# The Enchilada authenticated ciphers, v1

## Abstract:

*This paper proposes a pair of authenticated ciphers for the CAESAR competition. They are based on the idea of using a stream cipher and a block cipher together to produce a hybrid cipher that is demonstrably resistant to all the usual attacks on either block ciphers or stream ciphers.*

*The specific proposals here are two hybrid ciphers using components known to be efficient and thought – after considerable analysis – to be secure. Enchilada-128 uses 12-round ChaCha and Rijndael with 10 rounds and 128-bit block size; the actual encryption is identical to AES-128, though the key schedule is changed and whitening added. Enchilada-256 is a high-security variant using 20-round ChaCha and Rijndael with 14 rounds and 256-bit block size. Both ciphers use a Galois field construction borrowed from AES-GCM mode to add authentication.*

*I give a security proof for this class of cipher. If that proof is valid, it shows that these ciphers have provable lower bound security of $2^{blocksize}$ under mild assumptions, even if both ChaCha and Rijndael should turn out to be remarkably weak.*

## Designer and submitter:

**Sandy Harris** **sandyinchina@gmail.com**

**2014-05-18**

# Table of Contents

# Specification

Enchilada-128 and Enchilada-256 are hybrid ciphers using the ChaCha stream cipher and a whitened form of the Rijndael block cipher. This design has both advantages and disadvantages.

- The major advantage is a provable lower bound on security against a wide range of attacks. The bound is $2^{blocksize}$, so $2^{128}$ or $2^{256}$ for the particular ciphers proposed here.
- Both ciphers use a 256-bit key so the bound for a brute force attack is $2^{255}$; this means Enchilada-256 does not quite achieve 256-bit security, but the difference is insignificant.
- The authentication uses the Galois field operations from AES-GCM; the Enchilada ciphers are quite similar to that mode and have most of its advantages. The differences are in the way some parts of the authentication are initialised.
- Since Enchilada uses both a stream cipher and a block cipher, it is inherently slower than a scheme that requires only one of those. In particular, Enchilada-128 is inherently slower than AES-GCM using AES-128. However, choosing fast ciphers in both roles does not compromise security, so overall the scheme can be made reasonably fast.
- The comparison may be more favorable in other cases. For example, Enchilada-128 uses a 256-bit key and is intended to offer high security, but a speed comparison with AES-GCM using AES-256 might go either way since one needs ChaCha while the other needs four extra AES rounds.

I believe the trade-offs are reasonable, in particular that having provable security justifies the costs, but there is considerable room for discussion.

## *Changes from version 1*

This is version 1.1 of the Enchilada specification.

This version matches the submitted code; version 1 does not. There are two important changes:

Danilo Gligoroski pointed out on the mailing list that Enchilada, along with several other candidates, used the secret message number in a way that violated the requirements in the CAESAR call. He was correct, and I thank him. I have fixed that problem; Enchilada now only uses the public message number, passing it to ChaCha as the nonce there. The public and secret message numbers are discussed in more detail in the section “Variants” below.

The original specification used a whitened variant of ECB mode – add whitening to plaintext, apply encryption, and add whitening to the ciphertext. This led to problems in the implementation because I had not thought through the question of padding when I wrote the original document. I could have invented a padding scheme or – far more sensibly – adopted one of the usual ones, but then the code would not have matched the original specification, which is a CAESAR requirement.

Instead, this version uses the whole Enchilada AES+whitening combination in counter mode. The code does not match the original partly-broken specification, but it does match this one. The advantages of this mode are discussed under “Features” below.

## *Basic idea of the cipher*

It is possible to use a stream cipher and a block cipher together to derive a cipher that is, in some senses, stronger than either. This design is a much revised and expanded version of a paper [1] I did in 2008. The main new things are that I now have a security proof and I have added authentication.

All conventionally-designed block ciphers share a common theoretical weakness, because they all encrypt multiple blocks with a single key.

- An attacker may be able to collect many blocks encrypted with the same key, which allows a whole range of attacks that are impossible against a smaller number of blocks. Also, if he or she can extract a key, then he or she breaks all other blocks encrypted with that key at neglible cost.

The surge of interest in tweakable block ciphers over the past few years can be seen as at least partly an attempt to address this problem.

All conventionally-designed stream ciphers share a common theoretical weakness, because they all mix plaintext and keying material in a simple way. A typical stream cipher has two parts, a keyed pseudorandom number generator and a combiner which mixes pseudorandom material with plaintext to produce ciphertext. Typically, the combiner is just a bitwise XOR.

- Using XOR – or any other simple invertible operation – has an inherent weakness; it is *trivially* easy for an enemy to acquire a sample of the keystream generator's output, given some known or guessed plaintext. That is, the first step in attacking the generator is essentially free.

In general, neither of these weaknesses is important in practice; ciphers are designed to resist attacks even when the enemy has large samples of known plaintext, or even when he can choose plaintext or ciphertext. In particular, stream cipher generators are designed to resist attacks even when the enemy has a large sample of their output, and block ciphers are designed to resist differential and linear attacks in which the enemy has enormous amounts of data.

It can be argued that these should not be called “weaknesses”, that they would be better described as just natural properties of block ciphers and stream ciphers. I reject that argument. These properties are indeed quite natural, but they are not *necessary*; it is entirely possible to design a cipher that has neither property.

Specifically, a hybrid cipher such as Enchilada does not have these problems.

- If stream cipher output is mixed into the block cipher state – whitening in Enchilada, but round keys or a tweak could be used instead – then an enemy cannot get multiple blocks identically encrypted, so **most of the usual attacks on a block cipher** are blocked.
- From the stream cipher point of view, using a block cipher as the mixer blocks **most of the usual attacks on a stream cipher** – all the ones that require direct access to generator output.

That is, this class of hybrid is inherently resistant to most of the usual attacks on either block ciphers or stream ciphers. Later sections go into more detail.

## *Components*

The components used are ChaCha, Rijndael and the authentication mechansim from AES-GCM; all three are efficient, all have been extensively analyzed, and all are thought to be secure.

**AES** is a US government Federal Information Processing Standard (FIPS 197 [2]), the winner of a competition (historical information [3]) run by the National Institute of Standards and Technology (NIST) to find a suitable algorithm. Multiple implementations in both hardware and software and many papers on the cipher are available from various sources. **Rijndael** [4] is the more general cipher from which AES was derived; it is defined for 128-, 192- or 256-bit blocks but only the 128-bit variant is used in the AES standard.

Using Rijndael as a component means we can take advantage of available optimised code, of AES-specific instructions now built into some CPUs, and of the extensive existing analysis of AES. However, this is not entirely straightforward since we are not precisely using AES. Both Enchilada variants use a different key schedule – ChaCha provides the round keys – and Enchilada-256 uses a different block size, so additional analysis is needed.

A positive effect of this difference is that the altered key schedule blocks related-key attacks which depend on the standard AES key schedule. This is important since several attacks of this type have been published; references are given in the "Security analysis" section.

Using ChaCha to provide round keys appears reasonable since a stream cipher generator and a block cipher key schedule do essentially the same thing – expand a relatively small key into a larger set of apparently random keying material – and the stream cipher has to do more of that with more direct risk of the output being exposed to an enemy, so it must be designed to tougher criteria and therefore can be expected to cope well with the easier role as a key schedule.

**ChaCha** or Snuffle 2008 [5] is an updated version of Salsa20 or Snuffle 2005 [6] from the same designer, Daniel Bernstein. Salsa was one of the winning algorithms in the European eSTREAM competition [7]; specifically, it was chosen for Profile 1, ciphers suitable for software implementation. It did not win in the hardware category, but the designer says it is suitable for that use:

> Salsa20/8 and Salsa20/12, like the original Salsa20/20, offer a wide range of attractive options for the hardware implementor. They can fit into a very small circuit area; alternatively, they can be parallelized for extremely high throughput ... [8]

Both Salsa and ChaCha have variants with different numbers of rounds. The eSTREAM site has:

> Within eSTREAM, three main variants of Salsa20 - depending on the number of rounds r - were proposed: Salsa20/8, Salsa20/12 and Salsa20/20. Each provides a different security vs. performance trade-off. Salsa20/20 is recommended by the designer for "encryption in typical cryptographic applications". The versions Salsa20/12 and Salsa20/8 have 12 and 8 rounds, respectively, and the designer recommends them for "users who value speed more highly than confidence". The eSTREAM committee suggested the use of Salsa20/12, as offering the best balance among the different versions, combining very good performance with a comfortable margin of security. [9]

I refer to the three variants of ChaCha as ChaCha8, ChaCha12 and ChaCha20 according to the number of rounds. I use ChaCha12 in Enchilada-128 and ChaCha20 in Enchilada-256. Other ChaCha variants could be considered as possible tweaks.

**AES-GCM** [10] is Galois Counter Mode. It has been standardised by NIST [11] and is applied in RFCs 4106 and 4543 for IPsec, 5288 and 6367 for TLS and 5637 for SSH. I use its authentication code, based

on multiplication in a Galois field. I use ChaCha output for the hash key, H, rather than an AES encryption of the all-0 plaintext.

## *Variants*

There are two variants of my algorithm for somewhat different uses. Since "salsa" can refer to a condiment as well as a dance, I read it that way and name my cipher Enchilada after a dish that contains salsa plus other ingredients.

- My primary proposal, Enchilada-128, uses ChaCha12 and Rijndael with 10 rounds and 128-bit blocks. This Rijndael variant is identical to AES-128 except for the key schedule. The combination gives provable $2^{128}$ security and can straightforwardly take advantage of existing code or hardware for fast AES implementations.
- Enchilada-256 is a higher security variant. It uses ChaCha20 and Rijndael with 256-bit blocks and 14 rounds. This gives provable $2^{255}$ security. This can also benefit from work on optimising AES, but the process is less straightforward and may not succeed in all cases.

Large parts of Enchilada have already been recommended by standards bodies. AES-128 is recommended by NIST and is extremely widely used; Enchilada-128 uses exactly the same actual encryption though the key schedule is different and whitening is added.

Salsa20/12 was recommended by the eSTREAM project and ChaCha12 is an updated version from the original designer. Salsa20/20 was his recommendation for a conservative choice, and ChaCha20 is his updated version of that.

The security proof given later applies to both variants. If that proof is valid, then both variants have a ***provable lower security bound of their Rijndael block size*** against a wide variety of attacks, even if both ChaCha and Rijndael should turn out to be severely flawed.

That is, they provably meet the security goals shown in the "Goal" column below.

| **Enchilada variant** | | **ChaCha** | | **Rijndael** | |
|---|---|---|---|---|---|
| **Name** | **Goal** | **Rounds** | **Key size** | **Block size** | **Rounds** |
| Enchilada-128 | 128 | 12 | 256 | 128 | 10 |
| Enchilada-256 | 255 | 20 | 256 | 256 | 14 |

A brute force attack against the 256-bit ChaCha key breaks either Enchilada variant at cost $2^{255}$. My proof of difficulty, based on the Even-Mansour bound, gives $2^{\text{blocksize}}$ against many attacks but does not apply against a brute force attack. Overall difficulty is then the minimum of blocksize and keysize-1, so 128 and 255 for the two variants.

The security goal implies a minimum key size for ChaCha. I recommend a 256-bit key for both variants since all ChaCha variants support that. However, all variants of ChaCha also support 128-bit keys and using a 128-bit key with Enchilada-128 might make sense in some circumstances. For discussion see "Tweaking and tuning" below.

The block size determines the number of Rijndael rounds, as specified in the Rijndael paper.

In both variants stream cipher output is used to initialise the AES round keys. Setting up round keys this way blocks any attack that depends on the usual Rijndael or AES key schedule; this is significant since many of the published attacks on AES are related-key attacks.

Whitening is applied with addition; since both the AES key mixing and the authentication operations use XOR, this gives a bit of extra nonlinearity. 32-bit addition is used throughout because that is fast on nearly any CPU while 64-bit operations may not be; this adds a little extra overhead on some machines but saves some on others. The cipher uses little-endian additions throughout; this also implies extra overhead on some machines, but it does not appear excessive.

The CAESAR call [12] allows, but does not require, a fixed-size public message number, a fixed-size secret message number, and variable-size additional data. I provide for a 64-bit public message number, which is used as the 64-bit nonce for ChaCha, and for additional authenticated data; there is no secret message number.

If required, a calling protocol could reasonably easily add a secret message number of any convenient size. Simply generate a random number of suitable size, prepend or append it to the plaintext, and proceed with Enchilada encryption of the enlarged plaintext. The authentication would then include the secret number and decryption would give it along with the actual plaintext. Enchilada itself, however, does not provide this facility.

**Reusing the same combination of key and nonce would be fatally insecure** since it would cause the two instances of ChaCha to give identical outputs and all the other cipher state – Rijndael round keys, whitening values, counter initialisation and some values used in authentication – is produced by ChaCha. In this application, ChaCha and whitened counter mode AES are combined to give something that is effectively a stream cipher. Any such construction – or even a one-time-pad – becomes *horribly* vulnerable if the same key is used twice for different texts.

At the very least, the public message number should change, and ChaCha be rekeyed, for every invocation of Enchilada. There is further discussion under “Tweaks and tuning”.

The associated data is handled as in AES-GCM; the authenticator is run over it before starting on the ciphertext.

## *Authentication*

Both versions of Enchilada are similar to AES-GCM mode, with the whitened Rijndael replacing AES in the construction.

The main authentication operations on ciphertext, and the handling of length parameters for plaintext and additional data, are identical to AES-GCM; in fact they use code from Bernstein's AES-GCM implementation. Initialisation and finalisation of the authenticator are different; they use data from ChaCha instead of from encryption of an all-zero AES block. The effect is the same; the data is pseudorandom and key-dependent.

Enchilada-256 uses the same authentication as Enchilada-128 or AES-GCM; it just treats each 256-bit ciphertext block as two 128-bit input blocks for the authentication.

## *Tweaking and tuning*

There are a number of ways in which Enchilada could be changed to make it faster or stronger, or more convenient in some applications; this section discusses the ones that might be applied as tweaks in later rounds of the competition. Possibilities which I rejected for Enchilada are discussed later under "Design rationale".

## More or fewer ChaCha rounds

Attacking the stream cipher in this construction is difficult because (at least for the obvious attacks) an enemy has to first get through Rijndael and the whitening before he can acquire the data needed to go after ChaCha. Rijndael is believed to be strong and adding whitening must make it at least somewhat stronger. Having the whitening change cannot make it worse and appears to improve it again, so the block cipher part of the hybrid looks very strong. Later discussion will make this claim more precise.

On the other hand, if my security analysis is sound, then getting past the whitening would still cost at least $2^{blocksize}$ effort even if Rijndael were completely broken.

Given those considerations, having a strong stream cipher does not seem to be a critical requirement. That is, whatever stream cipher we use will be hard to attack and the security of the stream cipher is not critical for overall security. Despite that, I have made the moderately conservative choices of ChaCha12 for Enchilada-128 and ChaCha20 for the high-security Enchilada-256.

Changing the ChaCha variant is an obvious possibility for tuning; the number of ChaCha rounds in either Enchilada variant might be reduced or that in Enchilada-128 increased. In principle, it would be possible to offer six variants with two choices of Rijndael block size and three for the number of ChaCha rounds, or even to add more. However, offering such choices seems unnecessary; two variants aimed at 128-bit and 256-bit security seem sufficient, and both should use the number of rounds recommended by the Rijndael designers for their block size. The only open question, in my view, is what number of ChaCha rounds is appropriate for each.

I am very interested in analysts' comment on these questions, in particular on whether reducing the number of ChaCha rounds would be safe. I want to do that for performance reasons but do not feel confident doing it without analysis.

## Different nonce handling

The handling of the nonce might be adjusted to better match what higher-level protocols using authenticated encryption provide. The current code expects a 64-bit public message number which it uses as the 64-bit nonce for ChaCha. This is safe provided that the public number always changes and ChaCha uses its nonce effectively.

The ChaCha design document specifies a 64-bit nonce, and I follow that usage here. However, there has been work in the IETF on using it with a 96-bit nonce which is a better fit for some of the higher-level protocols. At time of writing, this is only at the Internet Draft stage as far as I can see, so I have not incorporated it here. If required, Enchilada could easily be tweaked to take that approach.

## 128-bit keys

The security goal implies a minimum key size for ChaCha. I recommend a 256-bit key for both variants since all ChaCha variants support that. However, all variants of ChaCha also support 128-bit keys, so they could be used here as well. The relevant table is:

| **Enchilada variant** | | **ChaCha** | | **Rijndael** | |
|---|---|---|---|---|---|
| **Name** | **Goal** | **Rounds** | **Key size** | **Block size** | **Rounds** |
| Enchilada-128 | 128 | 12 | 256 | 128 | 10 |
| Enchilada-256 | 255 | 20 | 256 | 256 | 14 |

For Enchilada-256, 128-bit keys would be an *exceedingly bad idea* since they would reduce security against a brute force attack to $2^{127}$, far below the goal for that cipher. This should not even be considered; it would be remarkably foolish to choose a high-security high-overhead cipher and then do something that drastically reduces its security.

For Enchilada-128, I do not recommend 128-bit keys, but they could be used without reducing security below that cipher's goal, or at least not significantly below; the difference between a $2^{128}$ goal and a $2^{127}$ possibilty for brute force attack does not matter much.

Adding support for 128-bit keys in Enchilada-128 would be trivially easy since ChaCha supports 128-bit keys. It might be required for some protocols, in which case it would be a straightforward tweak. Unless such a case is pointed out however, I believe it is both simpler and more sensible to support only 256-bit keys.

## *Pseudocode*

As a guard against ambiguity, I specify parts of the algorithm here in pseudocode. This is generic code which applies to both variants.

- BLOCK_BYTES is the Rijndael block size in bytes

Set up the key and nonce for ChaCha:

- memcpy( cha_cha_key, enchilada_key, 32 )
- memcpy( cha_cha_nonce, public_message_number, 8)
- run the ChaCha key schedule

For round key setup and the whitening, we use chunks of ChaCha output 128 bits at a time. The algorithm for getting these is:

- take 128-bit chunks from ChaCha output successively, starting from the lowest memory address
- run an iteration of ChaCha to get 512 new bits whenever we run out of inputs

Implementers may of course use larger chunks wherever that is convenient and gives the same result. The data is copied the round key array in the same order, lowest to highest memory address. In my code, I use the C memcpy() library function; anything else used should work the same way.

Using those methods, we do initial setup:

- set up two 128-bit values, H and I, for the authenticator using material from ChaCha
- initialise the counter for counter mode with ChaCha output
- set up the round keys by copying ChaCha data into them, in numeric order by their position in the round key array

The whitened AES is:

- get whitening data from ChaCha
- mix it into plaintext, using 32-bit additions
- apply Rijndael encryption to the counter
- mix the whitening into output, using 32-bit additions
- increment the counter (see the function white_aes() in source for complications)

The actual encryption is just an XOR of data from whitened AES into plaintext to give ciphertext. Since XOR is its own inverse, decryption works the same way.

The authentication is mostly exactly as in AES-GCM. The differences are:

- it uses Enchilada ciphertext
- H comes from ChaCha rather than an AES encryption of an all-zero block
- the final XOR creating the authentication tag uses I, also from ChaCha

Enchilada-256 treats each 256-bit ciphertext block as two 128-bit blocks for authentication.

## Security goals

For the encryption, the security goals are $2^{128}$ or $2^{255}$ effort for any attack.

The security proof given later applies to all variants. If that proof is valid, then ***all variants have a provable lower security bound of their Rijndael block size*** against a wide variety of attacks, even if both ChaCha and Rijndael should be discovered (to widespread surprise) to be horribly flawed.

Enchilada uses different Rijndael block sizes to achieve different security levels.

| **Enchilada variant** | | **ChaCha** | | **Rijndael** | |
|---|---|---|---|---|---|
| **Name** | **Goal** | **Rounds** | **Key size** | **Block size** | **Rounds** |
| Enchilada-128 | 128 | 12 | 256 | 128 | 10 |
| Enchilada-256 | 255 | 20 | 256 | 256 | 14 |

The key sizes must match or exceed the Rijndael block size. In Enchilada-128, key size could be reduced to 128 bits without violating this condition. For Enchilada-256, any reduction of key size would reduce the overall security level.

The number of Rijndael rounds is chosen to match what Rijndael requires for the block size, so overall we get:

security goal = provable security level

= Rijndael block size (implies number of Rijndael rounds)

<= ChaCha key size

The bound does not imply any particular number of rounds for ChaCha, so those choices are more a matter of judgement or guesswork; see “Tweaking and tuning” for discussion. In that context, it is worth noting that any ChaCha variant is safer here than in general use; to obtain stream cipher output an enemy has to get through Rijndael, not just XOR.

# Security analysis

There are several reasons to expect the structure used here to be extremely secure:

- Whitening – in particular, whitening that changes – makes attacking the block cipher difficult.
- Using the block cipher – instead of a simple XOR – to mix stream cipher output into text makes attacking the stream cipher difficult. This is true even when the text in question is a counter; the "outer" stream cipher (XOR whitened AES output into plaintext) is not protected in this way, but the "inner" stream cipher (use ChaCha to generate whitening) is.
- The ciphers used as components – ChaCha and Rijndael – are thought to be secure, and the combination appears to be at least as strong as the better of the two.
- Even and Mansour give a general proof for the difficulty of attacking a whitening structure, XOR-permutation-XOR. Applied here, that gives a proof that any attack that first breaks the block cipher then acquires enough stream cipher output for an attack on that must require at least $2^{blocksize}$ effort.

Each of these is discussed in more detail below.

## *Simple analysis*

This section discusses the overall security properties of this general class of hybrid cipher. It does not attempt either to give detailed proofs or to apply the discussion to the specific cases of the Enchilada ciphers; those issues are covered in later sections.

## Attacks on the block cipher

Viewed as a block cipher, this type of hybrid ***cannot be weaker than the block cipher with whitening*** added, even if the stream cipher turns out to be weak. The only exception would be a case where the stream cipher was so bad that using its output for the round keys opened up a new attack on the block cipher. It seems quite safe to assume ChaCha is not that bad.

Moreover, the hybrid ***resists any attack that needs more than one block*** encrypted in exactly the same way. Many attacks on a block cipher require more than one block encrypted in an identical way; such attacks are all blocked by the whitening changes. The most general and powerful techniques – linear and differential cryptanalysis – generally require huge samples of data encrypted in the same way. These – or at a minimum, the usual variants of them – are blocked as well.

The only attacks possible against a single block are brute force, various sorts of algebra, or combinations of the two using algebra to constrain a search sufficiently that brute force can succeed. Here all algebraic attacks are blocked because, with the stream cipher inputs, there are too many variables for the system to be soluble.

Brute force does not appear to do an attacker much good. He cannot try brute force (with or without some algebra to make it easier) against the primary key for the block cipher – e.g. the 128-bit key for AES-128 – since none is used here; instead the stream cipher generates the round keys. Brute force

against the whole round key array is astronomically difficult, and worse when whitening is added. Brute force against the 256-bit ChaCha key is difficult as well.

Also, even if an attacker can get the block cipher round keys with some combination of algebra and brute force, every future block has whitening and the whitening changes continually. ***Even with a completely broken block cipher, then, the hybrid is still at least as strong as the stream cipher***.

The hybrid construction defeats many of the published attacks on AES. In one of the Twofish team's papers [13] there is a table giving eight known attacks that break reduced-round Rijndael (now AES) with 6 to 9 rounds with various amounts of effort. The lowest amount of chosen plaintext required is $2^{32}$ blocks, encrypted with one key. Against Enchilada, an attacker cannot get even two blocks without a change in the whitening data. Of the attacks in that paper, the one that breaks the most rounds is: "a related-key attack on 9 rounds of Rijndael with 256-bit keys that uses $2^{77}$ plaintexts under 256 related keys, and requires $2^{224}$ steps". Enchilada uses stream cipher output for the round keys, so any attack that depends on the AES key schedule fails; this includes many of the published attacks. [14] [15] [16]

## Attacks on the stream cipher

From the stream cipher point of view, enemy attempts to get generator output are blocked. He has to break – or at least somehow compromise – the block cipher before he can get any generator data. This ***resists any attack that requires access to generator output***. Since all the usual attacks on stream ciphers require that access, this is a significant improvement over any conventionally-designed stream cipher.

Even breaking the block cipher does the attacker no good in attacking the stream cipher. A hybrid cipher with a broken block cipher is no worse than a stream cipher that uses XOR. An attacker who breaks the block cipher can get the pseudorandom data whenever he knows or guesses the plaintext. So can an attacker who "breaks" the trivial XOR in a normal stream cipher. That puts him in a position to attack the pseudorandom number generator, but it does not give him an attack on the generator.

That is, ***even with a completely broken block cipher, the hybrid is still at least as strong as the stream cipher***. Later in the paper, I shall prove that it is actually significantly better than that.

## Combined attacks

At least in this simple analysis, the hybrid construction appears to resist all the common attacks on either block ciphers or stream ciphers. There is no obvious way to attack the stream cipher without first compromising the block cipher and, on the other hand, mixing in some stream cipher output complicates any attack on the block cipher and completely prevents some attacks.

Discussion above shows that, excluding pathologically weak ciphers:

- even if the stream cipher is weak, the combination is at least as strong as the block cipher
- even if the block cipher is weak, the combination is at least as strong as the stream cipher

Combining these, we have that ***the combination is at least as strong as the better of the two component ciphers***.

This simple combination appears to defeat all the common attacks on either block ciphers or stream

ciphers, short of brute forcing the key. However appearances can sometimes be deceiving and the arguments above fall well short of proving the security. We need a more thorough analysis, and the following sections give it.

### *The Even-Mansour construction*

For stronger arguments, we turn to Even and Mansour [17]. They analyzed the XOR-permutation-XOR sequence and proved a lower bound for it; for n-bit blocks and D known or chosen plaintexts, the time T to break it is bounded by $DT >= 2^n$.

There have been many papers exploring variations of this scheme or proposing attacks on the original or variants [18] [19] [20] [21] [22] [23] [24]. For Enchilada, the critical consideration is that none of these invalidate the Even-Mansour bound.

A Dunkelman, Keller and Shamir paper [25] has "we can show that the original two-key Even-Mansour scheme ... can be simplified into a single key scheme with half as many key bits which provides exactly the same security"; this justifies Enchilada's use of a single-key scheme. The same paper has "It is clear that the lower bound security proof of EM holds without any change for AEM" where "AEM" is Addition Even-Mansour. This justifies the use of addition to apply the whitening in Enchilada.

In a hybrid cipher with the whitening material changing often we clearly have a small value of D so T, the effort required break it, is close to the block cipher's block size. In a simple analysis of Enchilada, we have D = 1 since we change the whitening for every block, and ***the bound $T >= 2^{blocksize}$ follows immediately***.

In a more precise analysis, that bound holds in many cases even when the attacker has more than one block to work with. Later sections discuss both those cases and the ones where I cannot prove the bound. I regard my work there as preliminary and expect quite a bit of comment from analysts.

Note that if there is any cost at all for finding the whitening material, that is a qualitatively better result than one gets with a conventional stream cipher. With a conventional stream cipher, known plaintext gives keystream at negligible cost. In any hybrid cipher of this general type there is a cost and, while that cost may not always be prohibitive, it will never be negligible unless the block size is tiny. With substantial block sizes, as used in Enchilada, we get provably adequate security.

## Properties

The Even-Mansour construction has a number of properties which make it very attractive.

Critically, **the Even-Mansour bound is a *lower* bound on attack complexity**. That means that – provided we can work out how to apply it to a real cipher – we should be able to derive a lower bound for the cipher. This would give an ideal result; a cipher with a provable minimum security level.

*Upper* bounds are easy to get. For example, there are two trivial upper bounds for any block cipher with key size k and block size b. Brute force search breaks it with two blocks of known plaintext at average cost $2^{k-1}$ encryptions. A codebook attack breaks it completely with $2^b$ known blocks and weakens it significantly from around $2^{b/2}$. There is also a third upper bound set by the effort necessary for an algebraic attack. Generally it is not clear what this bound might be; we do not have a good estimate of it for most ciphers, just an intuition that it is huge. A stream cipher has a similar set of easily

found upper bounds.

Upper bounds, however, are not really all that useful; they merely give us something to aim at. For example, we try to design a cipher so that no other attack is better than brute force. If we succeed – or at least have good grounds to think we have succeeded – then we can use the brute force cost as a lower bound and argue that the cipher has that level of security. Such arguments are common and often useful, but they all depend on something that is basically sleight-of-hand, using an upper bound on attack cost as if it were a lower bound on cipher strength.

Where possible, it is obviously preferable to choose one cipher component which has a proven lower bound and then use it in such as way that it becomes possible to ***derive a provable lower bound for attacks on the overall cipher.*** That is what I do here, starting from the Even-Mansour proof.

Secondly, **the Even-Mansour bound is extremely general**.

There has been considerable work done attempting to show provable security (lower bounds on attack cost) against linear and differential cryptanalysis. Well-known methods are in the CAST papers [26], Knudsen and Nyberg [27] and Vaudenay's work on decorrelation [28]. There may be more that I am not aware of.

The Even-Mansour bound is very general. One summary is:

> *In their original paper, Even and Mansour formally proved that any attack must satisfy $DT>\Omega(2^n)$. The lower bound proof is information theoretic, and is applicable both to known plaintext attacks and to chosen plaintext attacks.* [29]

This generality is highly desirable since it is fairly common for ciphers proven secure against one type of attack to fall to some attack that goes outside the assumptions used in the proof. For example, the Knudsen-Nyberg construction was broken by a new type of attack [30] and the AES candidate DFC (Decorrelated Fast Cipher) [31], applying Vaudenay's theory, fell to a variant of differential analysis [32].

The Even-Mansour proof seems general enough that such a catastrophe is unlikely. However, I am not absolutely certain here so ***I encourage analysis on this point***; please concoct a catastrophe if you can.

Finally, the proof of the Even-Mansour bound **does not require that the block cipher be at all strong.** In fact, it assumes exactly the opposite; the enemy is assumed to have access to oracles that give him the output of the permutation for any input or the input for any output. Applied to a block cipher, that assumption is equivalent to assuming the cipher is completely broken.

The proof only requires a permutation that is in a specific sense randomly chosen. Any block cipher with a random key meets this requirement; in fact a common formal model of a block cipher, going all the way back to Shannon [33], is as a set of permutations with the choice of key selecting one of them. Any block cipher giving ciphertext the same size as plaintext must be a permutation or it cannot be decrypted. The choice of permutation should depend on the key in some way that is not obvious to a cryptanalyst, so it must be at least weakly pseudorandom.

We can easily go further, selecting a block cipher that is provably pseudorandom. Luby and Rackoff [34] prove that, given a suitable round function, a Feistel cipher is pseudorandom after three rounds and strongly pseudorandom after four. Other papers extend the proof to other types of cipher. [35] [36]

Using any reasonable block cipher with the full number of rounds, we can safely assume it is strongly

pseudorandom. That is any block cipher, except perhaps for a spectacularly bone-headed design, meets the requirements of the Even-Mansour proof. Beyond question, Rijndael meets the requirements for the permutation in that proof and would continue to do so even if some amazing attack were found that rendered it useless as a general-purpose block cipher.

In Enchilada or similar hybrids, then, any lower bound proven with Even-Mansour analysis will hold even if the block cipher is very weak. In the actual design, I have of course tried to choose a strong block cipher, though large block size and efficiency were also considered. However, a security proof can be derived for this sort of hybrid without an assumption that the block cipher is strong.

## Bounds for Enchilada

The Even-Mansour bound for the whitened construction (with a broken block cipher and without changes of the whitening) is that for n-bit blocks and D known or chosen plaintexts, the time T to break it is bounded by $DT >= 2^n$. This section, and the following one, discuss how that applies to Enchilada.

In much of the discussion, I use the assumption from the Even-Mansour proof that the block cipher is already broken, that it adds no strength to the overall cipher and only acts as a permutation already known to the attacker. Of course this means that if – as seems extremely likely – Rijndael is actually at least reasonably sound, then Enchilada is in fact far stronger than this discussion shows.

For D = 1 we have $T >= 2^n$, so any attack on a single block must be at least that difficult. Arguably, this is the applicable case for any attack since the whitening uses new ChaCha output for every block, so each block is independent and any attack on whitening must use D = 1.

It is immediately clear that the bound holds for:

- any attack using only one known or chosen plaintext/ciphertext pair
- any attack that treats the blocks independently; since each block needs $T >= 2^n$ effort, the overall effort must also be above that bound
- any incremental attack that starts by solving one block then uses the result to attack others; since solving the first block is above the bound, the whole process must be

We also have – without even considering the Even-Mansour bound – that Enchilada is not vulnerable either to any of the usual attacks on stream ciphers since those require access to generator output or to most of the usual attacks on block ciphers since those require multiple blocks identically encrypted.

However, it is not clear that the bound holds for all attacks. Cases not covered above are:

- attacks using more than one known pair
- incremental attacks where the base case uses more than one pair

We need only prove the first; the second then follows as an immediate corollary.

## Attacks with large amounts of data

The key question is whether, given many blocks of known or chosen plaintext, some attack can be found against the overall cipher. The attacker naturally hopes for something more efficient than banging his or her head against the Even-Mansour bound. Of course my intent is that no efficient attack of this type exist.

An attacker may have reason for hope. The same basic strategy – collect many blocks of data and then do something clever with them – is used in a wide range of attacks on various ciphers and hashes, and is often quite effective. In particular, it has been used in cryptananlysis of many block ciphers and is the basis for some of the most powerful known general attacks, linear and differential analysis.

The variants of differential or linear analysis generally used against block ciphers obviously fail against Enchilada since they require many blocks identically encrypted. However, clever folk might construct other variants. Some of the details would need to be different in work against a hybrid cipher, but the same general method is clearly applicable, at least in principle.

This is therefore another area where ***I encourage analysis***. In fact, this is the area where I would be least surprised by news that someone had broken my cipher.

## A low lower bound

With no change of whitening and $D=2^d$ blocks of known or chosen plaintext we have $T >= 2^{n-d}$ from a direct application of Even-Mansour; this finds the two whitening blocks. That gives us a sort of lower bound on the lower bound; *even if the whitening never changed and Rijndael were completly broken*, the Enchilada construction would be at least that good. We also have an upper bound of $2^{255}$ for a brute force attack on the 256-bit key. The question is whether we can get tighter bounds, in particular a better lower bound.

Note that for use in real protocols, the $T >= 2^{n-d}$ bound may be enough. Consider IPsec as an example; an administrator can set a time limit, a limit on the amount of data encrypted, or both. The connection is automatically rekeyed whenever either limit on either gateway machine is exceeded. Then there is a fallback limit in the protocol; the connection is rekeyed after $2^{32}$ packets even if both adminstrators neglect to set any tighter limits. The usual packet size is under 2 Kbytes, so $2^{32}$ packets is typically under $2^{36}$ Enchilada blocks. In the normal case – where the administrators are awake and their policies sensible – rekeying will occur well before that.

Assume that Enchilada is rekeyed after $2^{40}$ blocks, so we have d = 40. Then $T >= 2^{n-d}$ gives $2^{88}$ for Enchilada-128 and $2^{216}$ for Enchilada-256. That is, even this bound is enough to show that, with any sane rekeying policy, Enchilada-128 is moderately secure and Enchilada-256 highly secure.

These are respectable numbers and straightforwardly provable, but their significance for Enchilada can be taken two ways. They can be taken as an argument for Enchilada since the bounds are good enough for some applications and we have good reason to suppose that either Enchilada variant will be considerably stronger than the corresponding bound.

On the other hand, since the same bounds can be proven for Rijndael alone (take the first and last round keys as whitening and the rest of the cipher as a permutation), they provide an argument that the

Enchilada construction is unnecessary. That argument has considerable force for Enchilada-256 since $2^{216}$ is almost certainly enough for practical purposes and is not all that far from the $2^{255}$ bound for a brute force attack. The argument is much weaker when applied to Enchilada-128.

## Bounds with the stream cipher

The fact that Enchilada regularly mixes in new stream cipher material gives reason for hope that the actual bounds are better. With the changing whitening and D encrypted blocks the attacker needs to find D blocks of whitening data plus the Rijndael round keys to break the block cipher part of Enchilada. If we call the size of the round key array R and the block size B, the attacker can get DB nonlinear equations and wants to solve for DB+R unknown bits. Given that R is non-zero, such a system of equations will remain underdetermined for any value of D.

This is completely different from a typical stream cipher using XOR where the equations for this part of an attack are fully determined and linear so, given some known plaintext, finding some generator output is trivial.

It is also quite different from the usual situation in an attack on a block cipher. With DES [37] there are only 56 key bits so even a one 64-bit known plaintext/ciphertext pair gives overdetermined equations. For a more modern cipher you need more blocks, but not a lot more. For example, AES-128 gives overdetermined equations with just two known pairs and AES-256 with three. The same relation holds for any of the other AES candidate ciphers. Even with independent round keys in AES, only twelve or sixteen pairs are needed.

Just yielding overdetermined equations does not mean the cipher can be broken in practice. Taking DES as an example, in principle an algebraic attack clearly breaks it but the attack does not appear practical. Intelligence agencies have both huge budgets and many clever people, DES was a very high-value target and they had decades to work on it, so it is at least conceivable that one or more such agency has a gadget that will take one known plaintext/ciphertext pair and more-or-less immediately output the key. However, nothing I am aware of – neither leaks from such agencies nor anything in the open literature – indicates that such devices exist or even that they could be built. The amount of computation required is enormous, apparently enough to be horrendously inconvenient even for a hardware implementation done with an intelligence agency budget.

Against DES, the best theoretical attacks are still linear [38] or differential [39] analysis; both need $2^{40}$ or more known plaintext blocks. In practice, the cheapest technique is brute force key search which needs $2^{55}$ effort but is relatively easily split into many parallel tasks on a suitable machine [40] [41].

For AES, both the number of equations and the size of the round key array are considerably larger and the key schedule is more nonlinear than DES so such a device looks astronomically impractical, even though it is clearly possible in theory. However, in principle any overdetermined system is soluble if the equations are linear and a sufficiently overdetermined system (just add more pairs) becomes soluble even if the equations are nonlinear.

Rijndael has a lovely algebraic structure [42] and it seems conceivable that some clever attack exploiting that structure could be found; indeed, Courtois and Pieprzyk [43] claim to have already found principles that could lead to such an attack on AES, and there are other papers along similar lines [44] [45] [46]. This work is theoretical; while it is widely acknowledged as interesting, no actual attack based on it has been

proposed.

Courtois' web page on AES has:

> Schneier and Ferguson in their recent 2003 book "Practical Cryptography" say about AES: *we don't quite trust the security...No other block cipher we know of has such a simple algebraic representation. We have no idea whether this leads to an attack or not, but not knowing is reason enough to be skeptical about the use of AES*.

For Enchilada, any theoretical weakness of Rijndael against algebraic attacks becomes irrelevant. The fact that the systems of equations for Enchilada are always underdetermined means that – provided the stream cipher output is effectively random from the attacker's point of view – there cannot be a way to solve them without guessing variables or iterating through the possibilities. The degree to which they are underdetermined, R, is huge so the work factor for such an attack, $2^R$, is enormous.

This lets us take our earlier analysis:

- Whitening – in particular, whitening that changes – makes attacking the block cipher difficult.

and make it much stronger:

- No attack that works only on the block cipher part of the hybrid can succeed without enormous cost using any amount of known plaintext if it tries to find both whitening and round keys.

In a way, this is a remarkably uninteresting result; all it says is that if the stream cipher is secure, then the hybrid is too. From another viewpoint, it is a very strong assertion – the block cipher is *effectively invulnerable* to any attack except one that also exploits some weakness in the stream cipher.

## Summary of security claims

Combining the results from various sections above, we have:

- Enchilada is not vulnerable to any of the usual attacks on stream ciphers since those require access to generator output.
- Enchilada is not vulnerable to any of the usual attacks on block ciphers since those require multiple blocks identically encrypted.
- The ciphers used as components – ChaCha and Rijndael – are both thought to be secure, and the combination appears to be at least as strong as the better of the two.
- No attack that works only on the block cipher part of the hybrid can succeed without enormous cost using any amount of known plaintext if it tries to find both whitening and round keys.

We also have a proven bound, effort >= $2^{blocksize}$, for a number of cases:

- any attack using only one known or chosen plaintext/ciphertext pair
- any attack that treats the blocks independently; since each block needs T >= $2^n$ effort, the overall effort must also be above that bound
- any incremental attack that starts by solving one block then uses the result to attack others; since solving the first block is above the bound, the whole process must be

This is not a complete proof of security, but it does cover many interesting cases including all the usual

attacks on either stream ciphers or block ciphers and nearly all the obvious attacks on a hybrid.

The critical consideration is that – even if both the stream cipher and the block cipher are moderately awful – the enemy must still do at least one Even-Mansour computation with $2^n$ overhead to get some data to mount an attack on the stream cipher. Hence I conjecture – though I cannot prove it for all cases – that:

> **A hybrid of this type which applies whitening to a block cipher with n-bit block size has $2^n$ security even if both the component ciphers are weak.**

***This is the most interesting claim in this paper, and the one most in need of analysis by others.***

I do not claim the combination is secure if either component cipher is pathologically weak. For example, if the block cipher is entirely linear then the Even-Mansour proof is invalidated, and with a sufficiently awful stream cipher the enemy could get both the round keys and the whitening. However, I do claim the Enchilada construction would still be secure up to the Even-Mansour bound even if one or both component ciphers had a serious but not utterly catastrophic flaw.

## *Attacks I cannot rule out*

For a hybrid cipher combining a stream cipher and a block cipher in this way, and any attack that proceeds sequentially with the steps:

1) break the block cipher
2) collect whitening data, which is just the stream cipher output
3) attack the stream cipher

The analysis above shows that, under some more-or-less reasonable assumptions:

1) step 1 above is effectively impossible, requiring enormous effort
2) for step 2, we have a lower bound of $2^{blocksize}$ on the cost of finding the whitening used assuming the block cipher itself is broken. We can choose the parameters to make that lower bound large enough for more-or-less any security requirement
3) step 3 remains as difficult as for the underlying stream cipher

However, the assumptions may not hold and, even if they do, not all possible attacks are excluded. This section covers the attacks I am aware of that are not excluded; analysts are invited to create more.

As for any cipher, a brute force attack against the primary key is possible but for Enchilada the cost is $2^{255}$; this matches the security goal for Enchilada-256 and is far above that for Enchilada-128. Also as for any cipher, a poor method of choosing keys – in particular a weak random number generator – can hugely reduce the security and anything that that lets an enemy obtain the key instantly reduces it to zero, but these are problems for the system security architect rather than the cipher designer.

Variations on linear and differential cryptanalysis, and on algebraic attacks, have been used against both block ciphers and stream ciphers, so clearly an attacker might try to come up with some sort of combination attack that aims to break the stream cipher and the block cipher in a hybrid simultaneously. This looks difficult, but it is clearly not inconceivable.

## Exploiting a weak stream cipher

The analysis above includes "The fact that the systems of equations for Enchilada are always underdetermined means that – *provided the stream cipher output is effectively random from the attacker's point of view* – there cannot be a way to solve them without guessing variables or iterating through the possibilities." (emphasis added)

Actually, however, things may not always be so rosy. The analysis holds if the stream cipher output is indistinguisable from random from the attacker's point of view, but that is not necessarily the case. If the attacker knows some constraining condition on stream cipher output, then he or she may be able to construct an attack on the whole system.

Related-key attacks are, in theory, a serious concern since some such attacks against Salsa [47] [48] have been proposed. However, they do not appear to matter a great deal in practice since in most applications an attacker has no way to influence what keys are used. Without such influence, or a very poor key generation method, the chances of related keys being used are very small and an attacker has no way to discover any relation between keys without first breaking the cipher.

Only a truly appalling weakness in the stream cipher would directly break a hybrid with the Enchilada structure by giving away enough information on the round keys and whitening to make breaking the block cipher part easy; it seems safe to assume ChaCha is nowhere near that bad, so solving the block cipher will remain difficult even if some weaknesses are found in ChaCha. If some really remarkable attack on ChaCha were discovered it would be straightforward to replace it with a better stream cipher.

The real concern here is that even a relatively minor weakness in the stream cipher might contribute to a meet-in-the-middle attack, as discussed in the next section.

## Meet-in-the-middle attacks

The text above has "No attack that *works only on the block cipher part* of the hybrid can succeed without enormous cost ..." (emphasis added). Of course an attacker is not constrained to use such attacks; in fact he is likely to avoid them.

Some form of meet-in-the-middle attack, treating the whitening as the middle, may be an attractive option for an attacker. This neatly puts the stream cipher and the block cipher on opposite sides of the middle so they can be attacked separately.

A key consideration here is that a meet-in-the-middle attack requires that the keys on either side of the middle be independent. That is not the case in Enchilada – the ChaCha key and the Rijndael round keys are actually closely related – but there is no reason the attacker cannot treat them as independent if the attack requires that. In fact he may be able to have his cake and eat it, treating them as independent when that seems useful and using their relationship when that is useful.

There is a difficulty here for the attacker. If he wants to treat the ChaCha and Rijndael keys as independent, then on the Rijndael side he has a key the size of the entire round key array to deal with. This is at the very least inconvenient and it looks as though it should be prohibitively difficult unless there is some very large flaw in Rijndael.

If one or both ciphers are weak, some form of meet-in-the-middle could be quite problematic. Even if both are strong it might still be better to attack the two components in parallel than to attempt it

sequentially, first breaking the block cipher and then learning enough whitening data for an attack on the stream cipher.

### Partial knowledge of whitening

The text above has "No attack that works only on the block cipher part of the hybrid can succeed without enormous cost using any amount of known plaintext *if it tries to find both whitening and round keys*." (emphasis added) The enemy never has enough data to find both.

This does not rule out an attack that does not try to solve for both, but only aims at extracting some useful information on the whitening. With enough such information, an enemy might be able to mount an attack on the stream cipher generator. If such an attack succeeds completely, then the enemy can get the round keys and whitening and breaking the block cipher becomes trivial.

The attack need not succeed completely to be dangerous; if the enemy can extract enough information on the whitening to provide interesting constraints on the stream cipher, then he may be able to exploit those along the lines described under "Exploiting a weak stream cipher" above.

An enemy might also aim at just extracting information on the round keys, but this does not seem to be as serious a threat. Without whitening data, he cannot attack the stream cipher and even with all round keys known, Rijndael is still a permutation that meets the requirements of the Even-Mansour proof so getting whitening data is still difficult.

## Features

The key feature of this design is its **provable security**.

That makes the validity of my security proof the key question for analysis. If the proof is not valid, then the entire proposal becomes dubious. Enchilada may be fast enough, but it is almost certain not to be the fastest solution proposed. If it does not deliver the claimed provable security, then it almost certainly fails as a candidate. On the other hand, if the proof holds, I believe it is a strong contender.

Another important feature is that the proposal uses **well-analyzed components**. My method of combining the components is novel and therefore needs analysis, but other parts of the system are well understood. Rijndael, Salsa, counter mode and the Galois-based authentication have all had strong recommendations from important standards bodies, arrived at after extensive analysis. The other things used – like the idea of whitening and the XOR/addition alternation for nonlinearity – are also more-or-less standard components that have been previously analyzed.

Of course no component can be known with absolute confidence to be secure and my particular use of them can certainly be questioned. Also, I use ChaCha, which is not precisely Salsa, Rijndael with 256-bit blocks, which is not AES and not nearly as well analyzed as AES, and a 256-bit counter. Also, I apply the 128-bit authentication method with a 256-bit cipher. It is conceivable that these differences raise issues which previous analysis does not cover. Still, there is enough previous analysis to, at the very least, make Rijndael, ChaCha and the authentication plausible candidates for this use.

Counter mode is an **inverse-free construction**; it is not necessary to implement both encryption and decryption for the underlying block cipher, only one or the other. The most important payoff is that this can reduce the number of transistors required in a hardware implementation. However, it also has

advantages in software implementations; the code can be simpler and less space is needed for S-boxes or other tables.

Enchilada-128 is very similar to AES-GCM, except that it encrypts the counter with whitened AES rather than just AES, gets the multiplier H from ChaCha rather than from AES encryption of an all-zero block, and updates the counter differently so the Hamming weight changes more. This should have **all the desirable properties of AES-GCM except speed**. If my security analysis is valid, it may be worth trading off some speed to get higher security.

Another consequence of using well-known – indeed standardised – components is that Enchilada can **take advantage of existing work on optimisation**. This applies to some extent for ChaCha, Rijndael with 256-bit blocks and the authentication, but is especially significant for AES and perhaps even more so for AES-128 which, except for the key schedule, is exactly what Enchilada-128 uses. There has been extensive work on fast software for this on any system from 8-bit smartcard CPUs up and on hardware from FPGA implementations to special AES-oriented instructions now built into some CPUs.

Fast implementations of Rijndael, ChaCha and Galois authentication can be done in a variety of ways and this proposal leaves implementers free to choose any of them. However the code that fits them together, the actual Enchilada code that I have written, **uses only very simple operations**. Some of the setup code uses *memcpy()*, a standard C library function that should be reasonably efficient on any system. The mixing of the whitening into plaintext and ciphertext is done with 32-bit additions. No operations likely to be expensive on some systems – such as multiplication, division, modulo, rotations or other bit-fiddling with masks and shifts – are used. Also, my code **does not need a large state**, though both ChaCha and Rijndael need more.

There are places where an implementer can use one 64-bit XOR instead of two 32-bit ones if that saves time on a particular target system, but nowhere in Enchilada128 that any 64-bit operation is actually required. Enchilada256 updates the counter with 64-bit operations – this gives faster changes of Hamming weight and lets me reuse code from Enchilada128, changing only the data size – but those are the only 64-bit operations in either cipher.

A useful effect of using only simple operations is **resistance to timing-based attacks**. ChaCha also resists such attacks, see section 2 of the original Salsa paper [49].

The proposal uses Rijndael and ChaCha, and those appear to be fine choices. It is worth noting, however, that **the Enchilada structure is quite generic** and other ciphers could fairly easily be substituted into it.

- One could replace ChaCha with some other stream cipher and nothing else in the basic proposal – except perhaps details like the size of the nonce – would need to change.
- One could also change the block cipher used without having to change anything else except for the number and size of round keys. For example, it would be entirely straightforward to construct a variant of Enchilada-128 using Camellia [50] or ARIA [51] for the countries where one of those is the government standard.

Such changes do not appear immediately necessary to me and are not part of this proposal.

### *Parameter sets*

The CAESAR call includes the text “Each submission must include a prioritized list of recommended parameter sets: a primary recommendation, a secondary recommendation, etc.” My list has only two entries:

1. Enchilada-128 is the primary proposal, a general-purpose authenticated cipher offering security likely to be adequate in nearly all applications.
2. Enchilada-256 is a high-security variant for applications where extra security assurance is more important than low overheads.

The relevant table is:

| **Enchilada variant** | | **ChaCha** | | **Rijndael** | |
|---|---|---|---|---|---|
| **Name** | **Goal** | **Rounds** | **Key size** | **Block size** | **Rounds** |
| Enchilada-128 | 128 | 12 | 256 | 128 | 10 |
| Enchilada-256 | 255 | 20 | 256 | 256 | 14 |

## Design rationale

This section discusses some of the design choices made in Enchilada. The main design ideas have already been covered under “Basic idea of the cipher” and “Security analysis” above, so the focus here is on ideas I considered but rejected.

The CAESAR call requires the statement "The designer/designers have not hidden any weaknesses in this cipher." That is certainly true of Enchilada.

### *Rejected options*

There are a number of alternate design choices which I chose not to explore here. They might be used in other ciphers using the basic notion of building a hybrid from a stream cipher and a block cipher, but they seemed inappropriate for Enchilada.

It would be possible to **eliminate the separate whitening** data and directly manipulate the first and last Rijndael round keys instead; this would save the overhead of applying whitening before and after each encryption. This is a somewhat plausible approach for any block cipher, and particularly so for a substitution-permutation network such as Rijndael because an n-round SPN can be seen as a permutation, consisting of n-2 rounds plus some miscellaneous stuff, and two whitening keys. It would also be possible to devise schemes in which stream cipher data is used to alter all of the block cipher round keys. I rejected these alternatives here because separate whitening gives a cleaner, more easily analyzed, design.

One might **reduce the number of Rijndael rounds** for speed, arguing that with the whitening fewer rounds are needed, but I chose not to. Changing the number of ChaCha rounds, within the range specified by the designer, might be reasonable but for Rijndael it seems safer to stick with the numbers recommended by the designers.

One might also **increase the number of Rijndael rounds** for greater security. Since Enchilada-128 takes a 256-bit key, arguably it should use 14 rounds like AES-256 rather than 10 like AES-128. Similarly, one might increase the number in Enchilada-256; in fact one draft of this document included an Enchilada-20-20 variant with 256-bit blocks and 20 rounds each in ChaCha and Rijndael, but I removed it. For both variants, if my security proof holds then the extra rounds are unnecessary. If my proof fails, then better ciphers can almost certainly be designed by using some other approach rather than tweaking this one.

If someone finds the overall approach of hybrid ciphers sensible but thinks Enchilada-256 inadequate for some security requirement (to me, that is almost an oxymoron, but it seems conceivable), then he or she should not mess about with tweaks to the number of rounds in my ciphers. Nor should he or she change to another stream cipher or another block cipher with the same block size without good reason to suspect serious problems in ChaCha or Rijndael. The sensible approach would be to look at applying the Enchilada structure to larger block sizes; for example one might construct a 512-bit variant using the W cipher from the Whirpool hash [52], or use 512-bit or 1024-bit Threefish [53], or even larger ciphers. I have chosen not to do that here; it seemed unnecessary.

One might also **use less** than a block worth of **stream cipher output** per counter encryption, either using some smaller amount each time or changing the whitening completely after some number of encryptions larger than one. I rejected this approach, mainly because it would invalidate the argument that the cipher is immune to algebraic analysis.

One might use **independent keys** for the block cipher and stream cipher components of a hybrid, but the fact that a meet-in-the middle attack requires independent keys is a strong argument against this. Another is that some block ciphers are vulnerable to related-key attacks and using the block cipher key schedule will expose any weakness that exists there. Initialising the block cipher round keys with stream cipher output avoids that risk but might allow some other form of related key attack on the block cipher if the stream cipher had a major weakness. I do not consider this an issue for Enchilada, but analysis might prove me wrong.

If enough key material is available, the safest approach would be to first apply the block cipher key schedule and then XOR stream cipher output into the round key array; this is secure if either the key schedule or the stream cipher is. I did not do this in Enchilada since I did not see it as necessary. If reviewers do (which would surprise me considerably), it could be added as a tweak with the key schedule getting input either from an enlarged primary key or from ChaCha output.

Enchilada-256 uses the 128-bit authentication code from AES-GCM. Clearly it would be possible to devise a **256-bit authenticator** along the same lines; the main problem would be choosing an appropriate 256-bit polynomial. This would give a larger authenticator, perhaps a better match for the higher security encryption, and efficiency should not change much because it would use half as many multiplications each of twice as many bits. I did not do this because it seemed to entail more work, more chance of error in design or implementation, and more requirements for analysis without giving any clear benefit in most application contexts.

## *Questions for analysis*

I believe this is a somewhat novel type of cipher and I make some rather strong claims for it, so it may require or provoke more analysis and criticism than some other proposals. This section highlights the issues I can see, though analysts are of course welcome to raise others.

## The security proof

The design depends critically on the status of the security proof given. My strongest claims are that:

- Even if both the stream cipher and the block cipher are easily broken, the enemy must still do at least one Even-Mansour computation with $2^n$ overhead to get some data to mount an attack on the stream cipher.
- Therefore, a hybrid of this type which applies whitening to a block cipher with n-bit block size has provable $2^n$ security even if both the component ciphers are weak.

These are the most interesting claims in the submission, and the ones most in need of analysis by others. If they are demonstrably false, then the whole submission collapses; without the payoff of provable security, the cost of using both a stream cipher and a block cipher cannot be justified.

## How weak could components be?

If my security analysis holds, the combination is secure if neither cipher is horrendously weak. I claim that the Enchilada construction would still be secure up to the Even-Mansour bound even if one or both component ciphers had a serious but not entirely catastrophic flaw. I do not claim the combination is secure if either component cipher is pathologically weak.

Of course there is room for analysis of how weak each component might be without compromising the overall structure. This is not of enormous concern for Enchilada, since both ChaCha and Rijndael are thought to be strong, though it does affect whether using fewer rounds in ChaCha would be a reasonable tweak. However, it might be very important in the design of other hybrid ciphers of the general type.

One interesting question is whether it would be reasonable to **eliminate the stream cipher** part of the hybrid and just use a counter or some other simple device, such as a linear congruential generator in software or a simple shift register concoction in hardware. The HAIFA work [54] shows that mixing in a counter on each iteration of a hash compression function significantly improves security of the hash. Various other systems including the Skein hashes and the Salsa/ChaCha stream cipher also mix in a counter.

Might something similar work in the Enchilada structure? Even a simple counter is enough to defeat attacks which need many blocks identically encrypted if it is mixed into the block cipher state, either whitening or round keys, or is used as the tweak with a tweakable cipher. However, I did not consider this seriously for Enchilada; it seems to me a real stream cipher is required to make successive whitening values unpredictably different. Also, I am already using a counter for counter mode and it is not clear that using another elsewhere in the cipher would add value, or even that it would not introduce some vulnerability. Using simple fast stream ciphers such as ChaCha8 or Pike [55] appears to

me to be an avenue worth exploring, but I would not want to risk using something even simpler unless there was some very credible analysis showing it was safe.

Similarly, one might consider using **some sort of minimal block cipher**. The Even-Mansour proof requires only a randomly-chosen permutation, and any reasonable block cipher meets that criterion even with a radical reduction in the number of rounds. Some very simple constructions such as IDEA [56] multiplication of plaintext by key also meet it; would that plus a mixing transform to make the effective block size larger than 16 bits be enough?

For Enchilada, I chose to use a strong block cipher with the full number of rounds. I may explore other choices in a later paper, but for something I am proposing for standardisation it seemed prudent to be cautious in choosing components.

# Administrative things

The CAESAR call requires that certain sections be present. Here they are.

## *Intellectual property*

As far as I know, no part of this proposal is subject to any patent, patent application, trade secret or other intellectual property constraint. Nor do I expect it to be in the future.

The only part of my work here that may be original enough that it might conceivably be patented is the notion of combining a stream cipher and a block cipher as is done here. I have not attempted to patent that, nor do I intend to. I do not believe it could be patented in any case since I disclosed the main idea publicly in a 2008 paper. [57]

Comments by Terry Ritter in the Usenet sci.crypt newsgroup sometime in the late 90s influenced some of my thinking here – that was where I first encountered the idea that XOR is a less-than-ideal mixer for a stream cipher – though I think I have taken the notion well beyond Ritter's work. Some of his work was patented [58] and I suppose that a lawyer might make an argument that some of the patents apply here. I do not believe such an argument could be made successfully since the most relevant patent [59] is from 1989 so I think it is expired, and anyway the mechanism it describes is quite different from mine. However, I am neither a lawyer nor any other sort of expert in this area, so if the committee wants to be absolutely certain on the point it might be well to seek legal advice and/or to ask Ritter for some sort of assurance that he sees no claim.

I believe that the rest of the Enchilada design is unpatentable because it is standard stuff that should be known to anyone “skilled in the art”. I made a conscious effort to use only standard well-understood components, building on the work of others wherever possible, both because that saves work for me and because it facilitates analysis for everyone else.

I do use other people's work – Rijndael, ChaCha, and the authentication method. Those all contain original ideas which might be patentable, but Rijndael and ChaCha have already met stringent conditions with regard to intellectual property disclosure and release as part of the AES and eSTREAM competitions respectively and the AES-GCM paper mentions freedom from intellectual property restrictions as an explicit goal for that design.

I also use other people's code in my reference implementation.

For ChaCha I use one of Daniel Bernstein's implementations, the “merged” (unrolled) version from his ChaCha page [60]; the source states that this is public domain. My own code is also labelled in the source as released into the public domain.

Enchilada-128 uses the “low resource” AES code from Brian Gladmans' AES page [61] while Enchilada-256 uses code from his Rijndael page [62]. In both cases, Gladman specifies license conditions in the code and I have retained his text. I do not believe the license to be at all problematic, and if some problem were discovered it would be straightforward to substitute some other implementation.

I have not used Gladman's best-known AES code, the heavily optimised version also linked from his AES page, which is used in the Linux crypto libraries, in Peter Gutmann's Cryptlib and in many other places. That does not do 256-bit blocks which Enchilada-256 needs. For Enchilada-128, I chose to use

the simpler “low resource” code and simplify it further by assuming a 32-bit CPU though the original is designed to run even on 8-bit processors. To optimise either of my ciphers, or to make them run on very limited CPUs, Gladman's other variants would be an excellent first thing to look at. Bernstein also has other versions of ChaCha linked from his page above which would be a good starting point for anyone looking at optimizing or porting Enchilada.

The CAESAR call requires the statement:

> If any of this information changes, the submitter/submitters will promptly (and within at most one month) announce these changes on the `crypto-competitions` mailing list.

I agree to that.

## *Consent*

The CAESAR call requires that this section exist and that it contain the following text:

> The submitter/submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non-selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitter/submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitter/submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitter/submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitter/submitters understand that if they disagree with published analyses then they are expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitter/submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

I agree to all of that.

1 Sandy Harris, Exploring Cipherspace: Combining stream ciphers and block ciphers http://eprint.iacr.org/2008/473
2 FIPS standard for AES http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
3 NIST archive on the AES competition http://csrc.nist.gov/archive/aes/
4 Rijndael paper on NIST site: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf
5 ChaCha (Snuffle 2008) home page http://cr.yp.to/chacha.html
6 Salsa20 (Snuffle 2005) home page http://cr.yp.to/snuffle.html
7 eSTREAM cipher portfolio http://www.ecrypt.eu.org/stream/
8 Daniel Bernstein, Salsa20/8 and Salsa20/12, http://cr.yp.to/snuffle/812.pdf
9 eSTREAM page titled Salsa20/12 http://www.ecrypt.eu.org/stream/e2-salsa20.html
10 David A McGrew and John Viege, The Galois/Counter Mode of Operation (GCM) http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf
11 Morris Dworkin, NIST Special Publication 800-38D, Nov 2007, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
12 CAESAR call, draft four (Nov 2013) http://competitions.cr.yp.to/caesar-call-4.html
13 Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting "Improved Cryptanalysis of Rijndael", http://www.schneier.com/paper-rijndael.html
14 A. Biryukov and D. Khovratovich, "Related-Key Cryptanalysis of the Full AES-192 and AES-256", Asiacrypt 2009
15 A. Biryukov, D. Khovratovich, and I. Nikolic, "Distinguisher and Related-Key Attack on the Full AES-256", Crypto 2009, LNCS 5677, pp. 231-249, Springer-Verlag, 2009
16 Frederik Armknecht & Stefan Lucks, Linearity of the AES key Schedule, 4th AES Conference, 2004, http://th.informatik.uni-mannheim.de/people/armknecht/VortragAES.ps
17 S. Even, Y. Mansour, A Construction of a Cipher From a Single Pseudorandom Permutation, Asiacrypt 1991
18 Joan Daemen, Limitations of the Even-Mansour Construction, http://www.esat.kuleuven.ac.be/~cosicart/ps/JD-9102.ps.gz
19 Biham et al. "Cryptanalysis of Iterated Even-Mansour Schemes with Two Keys" *https://eprint.iacr.org/2013/674*
20 Craig Gentry and Zulfikar Ramzan "Eliminating Random Permutation Oracles in the Even-Mansour Cipher" http://link.springer.com/chapter/10.1007%2F978-3-540-30539-2_3
21 Orr Dunkelman and Adi Shamir, An Optimal Attack on Cryptosystems Using Pre/Post Whitening Keys, AC2010 rump session
22 Antoine Joux "Even-Mansour and the multi-users setting" *https://www.cryptolux.org/mediawiki.../Joux-EM-multiuser-ESC2013.pdf*
23 H Kuwakado "Security on the quantum-type Even-Mansour cipher"
24 Lampe, Patarin and Seurin, "An asymptotically tight security analysis of the iterated even-mansour cipher" http://dl.acm.org/citation.cfm?id=2437329
25 Orr Dunkelman, Nathan Keller, and Adi Shamir "Minimalism in Cryptography: The Even-Mansour Scheme Revisited" *https://eprint.iacr.org/2011/541.pdf*
26 Carlisle Adams, Constructing Symmetric Ciphers using the CAST Design Procedure, Designs, Codes and Cryptography, November 1997 http://cryptome.org/jya/cast.html
27 Kaissa Nyberg and Lars Knudsen, Provable security against a differential attack, Journal of Cryptology, 1995
28 Serge Vaudenay, Decorrelation: A Theory for Block Cipher Security, Journal of Cryptology, Springer, 2003
29 Dunkelmam, Keller and Shamir, slides http://www.cs.bris.ac.uk/eurocrypt2012/Program/Tues/Shamir.ppt
30 L.R. Knudsen, T. Jakobsen. The Interpolation Attack on Block Ciphers. FSE, 1997.
31 H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay, Decorrelated Fast Cipher: an AES candidate, May 1998, http://citeseer.ist.psu.edu/gilbert98decorrelated.html
32 Lars Knudsen & Vincent Rijmen, On the Decorrelated Fast Cipher (DFC) and Its Theory, FSE '99 http://www.cosic.esat.kuleuven.be/publications/article-367.ps
33 Claude Shannon, Communication Theory of Secrecy Systems, Bell Systems Tech Journal, 1949
34 M Luby & C Rackoff, How to construct pesudorandom permutations from pseudorandom functions, SIAM Journal on Computing, April 1988
35 Yiyuan Luo, Xuejia Lai, Zheng Gong & Zhongming Wu Pseudorandomness Analysis of the Lai-Massey Scheme, http://eprint.iacr.org/2009/266.pdf
36 Eric Miles & Emanuele Viola, Substitution-permutation networks, pseudorandom functions, and Natural Proofs, http://eprint.iacr.org/2011/226.pdf
37 FIPS 46-3 Data Encryption Standard, NIST http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

38 Matsui, M. "The first experimental cryptanalysis of the data encryption standard". *Advances in Cryptology - CRYPTO 1994*.
39 Biham, E. and Shamir, A. *Differential Cryptanalysis of the Data Encryption Standard - Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16–20, 1992, Proceedings*. 1992. p. 487-496
40 EFF DES Cracker, https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_descracker_pressrel.html
41 Sciengines website, Break DES in less than a single day, http://www.sciengines.com/company/news-a-events/74-des-in-1-day.html
42 Niels Ferguson, Richard Schroeppel, and Doug Whiting "A simple algebraic representation of Rijndael", SAC 2001, http://www.macfergus.com/pub/rdalgeq.html
43 Nicolas T. Courtois and Josef Pieprzyk "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations" http://eprint.iacr.org/2002/044
44 Sean Murphy and Matthew Robshaw, Essential Algebraic Structure within the AES, http://www.isg.rhul.ac.uk/~sean/crypto.ps
45 Harris Nover, Algebraic Cryptanalysos of AES: an overview, http://www.math.wisc.edu/~boston/nover.pdf
46 Carlos Cid, Some Algebraic Aspects of the Advanced Encryption Standard, http://www.isg.rhul.ac.uk/~ccid/publications/**aes**4_ccid.ps
47 Deike Priemuth-Schmid and Alex Biryukov, Slid Pairs in Salsa20 and Trivium, http://eprint.iacr.org/2008/405.pdf
48 Fischer, Meier, Berbain, Biasse & Robshaw, Non-randomness in eSTREAM candidates Salsa20 and TSC-4. In Rana Barua and Tanja Lange, editors, INDOCRYPT, volume 4329 of LNCS, pages 2–16. Springer, 2006.
49 Daniel Bernstein, The Salsa20 family of stream ciphers http://cr.yp.to/snuffle/salsafamily-20071225.pdf
50 Home page for the Camellia block cipher http://info.isl.ntt.co.jp/crypt/eng/camellia/index.html
51 Home page for the ARIA block cpher http://210.104.33.10/ARIA/index-e.html
52 Paulo Barreto and Vincent Rijmen, The WHIRLPOOL Hashing Function
53 Skein hash reference site http://www.skein-hash.info/
54 Eli Biham and Orr Dunkelman, A Framework for Iterative Hash Functions: HAIFA, http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2007/CS/CS-2007-15.pdf
55 Ross Anderson, On Fibonacci Keystream Generators, FSE3, http://www.cl.cam.ac.uk/~rja14/Papers/fibonacci.pdf
56 Xuejia Lai and James L. Massey, A Proposal for a New Block Encryption Standard, 1991 http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.3451
57 Sandy Harris, Exploring Cipherspace: Combining stream ciphers and block ciphers http://eprint.iacr.org/2008/473
58 List of patents for Terry Ritter http://patents.justia.com/inventor/terry-f-ritter
59 Ritter patent "Dynamic substitution combiner and extractor", Nov 1, 1989 http://patents.justia.com/patent/4979832
60 Bernstein's ChaCha page: http://cr.yp.to/chacha.html
61 Gladman's AES Page: http://gladman.plushost.co.uk/oldsite/AES/
62 Gladman's Rijndael page: http://gladman.plushost.co.uk/oldsite/cryptography_technology/rijndael/