Single-key AIL-MACs from any FIL-MAC*

Ueli Maurer and Johan Sjödin**

Department of Computer Science
Swiss Federal Institute of Technology (ETH), Zurich
CH-8092 Zurich, Switzerland
{maurer,sjoedin}@inf.ethz.ch

Abstract. We investigate a general paradigm for constructing arbitrary-input-length (AIL) MACs from fixed-input-length (FIL) MACs, define the waste as the relevant efficiency parameter of such constructions, and give a simple and general security proof technique applicable to very general constructions. We propose concrete, essentially optimal constructions for practical use, Chain-Shift (CS) and Chain-Rotate (CR), and prove their security. They are superior to the best previously known construction, the NI-construction proposed by An and Bellare: Only one rather than two secret keys are required, the efficiency is improved, and the message space is truly AIL, i.e., there is no upper bound on the message length. The generality of our proof technique is also illustrated by giving a simple security proof of the NI-construction and several improvements thereof.

Keywords: Message authentication code (MAC), arbitrary-input-length (AIL), variable-input-length (VIL), fixed-input-length (FIL)

1 Introduction

1.1 Message Authentication Codes (MACs)

Authenticity is a fundamental security requirement for data transmissions. A well-known technique for authenticating messages is to use a so-called *message authentication code* (MAC), an important symmetric-key cryptographic primitive with widespread use in many practical applications. A MAC is a function family $H := \{h_k : \mathcal{M} \to \mathcal{T}\}_{k \in \mathcal{K}}$, where $\mathcal{M}$ is the message space, $\mathcal{T}$ the tag space, and $\mathcal{K}$ the key space. Two communicating parties who share a secret key k can authenticate a message m, sent over an insecure channel, by computing a tag $\tau = h_k(m)$, which is sent together with the message. The message need not be encrypted. The receiver accepts the message if and only if the received pair (m', τ') satisfies $\tau' = h_k(m')$. We refer to h_k as an *instantiation* of H.

The message space is an important parameter of a MAC scheme. In most applications one needs to authenticate messages of potentially *arbitrary-input-length* (AIL), i.e., $\mathcal{M} = \{0,1\}^*$. Many proposed MACs are not AIL-MACs since

* To appear in Proc. ICALP 2005, LNCS 3580, pp. 472-484.
** This work was partially supported by the Zurich Information Security Center. It represents the views of the authors.

there is an upper bound on the message length, i.e., the message space is the set $\mathcal{M} = \{0,1\}^{\leq N}$ of all bit strings of length at most N. We refer to such MACs as having *variable-input-length* (VIL). The constant N is typically large enough to be of little practical concern. A MAC which has $\mathcal{M} = \{0,1\}^{L}$ for a constant L is referred to as having *fixed-input-length* (FIL). In this paper we address the problem of constructing (VIL- and) AIL-MACs from any FIL-MAC.

1.2 Previous Work

Constructing VIL- or AIL-primitives from FIL-primitives have been addressed in many papers. A well-known example is the Merkle-Damgård [5, 7] iteration method for constructing AIL collision-resistant functions from FIL collision-resistant functions. The question of constructing VIL- or AIL- pseudo random functions (PRFs) based on any FIL-PRF has received substantial attention, see for example the CBC-MAC [4, 6, 10] and the XOR-MAC [3] (which are PRFs and thus trivially also MACs). Other examples of AIL-MAC constructions are the hash function-based MACs like NMAC and HMAC [2].

A central goal in cryptography is to prove the security of cryptographic schemes under as weak assumptions as possible. In the context of constructing VIL- or AIL-MACs, a natural assumption (much weaker than the PRF assumption) is that the underlying FIL-primitive is a secure FIL-MAC. In 1997 Naor and Reingold [8] constructed a FIL-PRF from any FIL-MAC, but with high cost. While this FIL-PRF could in principle be used in some well-known construction of an AIL-MAC from any FIL-PRF (e.g. the CBC-MAC), it would be impractical. Their question whether a FIL-PRF can be obtained from any FIL-MAC at low cost is still open to date. In 1999 the problem of constructing VIL-MACs from FIL-MACs was proposed and investigated by An and Bellare [1]. They showed that the CBC-MAC is insecure under this weaker assumption for the FIL-primitive. They also presented the *nested iterated* (NI) construction, the first practical construction of a VIL-MAC based on any FIL-MAC (see Fig. 3). As we will see, the NI-construction leaves room for improvements.

1.3 MAC Constructions and Important Design Criteria

Throughout this paper, let $G := \{g_k : \{0,1\}^{L} \to \{0,1\}^{\ell}\}_{k\in\{0,1\}^{\kappa}}$ denote a FIL-MAC, with *compression* $b := L - \ell > 0$. We consider a general type of construction $\mathrm{C}^{\cdot}$, which uses G to construct a MAC $\mathrm{C}^{G} := \{\mathrm{C}^{g_k} : \mathcal{M} \to \{0,1\}^{\ell}\}_{k\in\{0,1\}^{\kappa}}$, where $\mathcal{M}$ is either AIL (i.e., $\{0,1\}^{*}$) or VIL (i.e., $\{0,1\}^{\leq N}$). The instantiation C^{g_k} is constructed by invoking g_k several times in a black-box manner. To be more precise, let us describe the computation of the tag $\tau = \mathrm{C}^{g_k}(m)$ for an n-bit message m. In a pre-processing step m is encoded into a bit string m' of length (denoted by) $\lambda(n)$, for instance by padding m and appending information about its length. The processing step is best described with a buffer initialized with m', where each call to g_k fetches (and deletes) some L bits and writes back the ℓ-bit result to the buffer. This reduces the number of bits in the buffer (by b bits) with each call to g_k. As soon as the number of bits is less than L, the content

of the buffer is returned as the tag τ. To obtain an ℓ-bit output, an appropriate encoding is used such that $\lambda(n) = t(n) \cdot b + \ell$ for some $t(n)$. Note that $t(n)$ is exactly the number of calls to g_k required to compute τ, and that τ is the last output of g_k. The function $t(\cdot)$ is referred to as the *application* function of $\mathrm{C}^{\cdot}$. A particular construction can thus be described by the encoding function mapping m to m' and by the scheme by which the L-bit blocks are fetched.

In a more general variant of such a construction, several (say 2) instantiations g_{k_1} and g_{k_2} of G can be used to build an instantiation $\mathrm{C}^{g_{k_1},g_{k_2}}$ of the MAC $\mathrm{C}^{G,G} := \{\mathrm{C}^{g_{k_1},g_{k_2}} : \mathcal{M} \to \{0,1\}^{\ell}\}_{k_1,k_2 \in \{0,1\}^{\kappa}}$ (with key space $(\{0,1\}^{\kappa})^2$). The only difference in the computation of the tag, described above, is that for each L-bit block that is fetched, the instantiation to be invoked needs to be specified. For such schemes $t^i(n)$ (with $i \in \{1,2\}$) denotes the number of calls needed to g_{k_i} in order to compute the tag of an n-bit message, and $t(n) := t^1(n) + t^2(n)$.

Note that the key space of $\mathrm{C}^{G,G}$ is twice the size of the key space of C^{G}. We refer to $\mathrm{C}^{\cdot}$ as a single-key construction and to $\mathrm{C}^{\cdot,\cdot}$ as a 2-key construction. We now discuss the main design criteria for the constructions:

Number of Keys: We will propose single-key constructions (like $\mathrm{C}^{\cdot}$) for practical use and see that there is essentially no reason for considering multiple-key constructions (like $\mathrm{C}^{\cdot,\cdot}$).

Efficiency: The efficiency can be measured in the number of *applications* $t(n)$ of the FIL-MAC, or equivalently in terms of the *waste* $w(n) := \lambda(n) - n = t(n) \cdot b + \ell - n$, i.e., the amount by which pre-processing expands the message.

Type of Processing: It is desirable that a message can be processed *on-line*, i.e., as the message bits arrive, without knowing the message length in advance. Moreover, it is desirable that the computation of the tag τ can be *parallelized*, i.e., sped up by a factor of roughly c (over the construction using one processor) when c processors are available.

Message Space: As we will see, it turns out that no bound on the message length is necessary, and therefore our focus is on AIL-MAC constructions.

1.4 Contributions of this Paper

The purpose of this paper is to investigate systematically a natural and general paradigm for constructing (VIL- or) AIL-MACs from FIL-MACs, a problem introduced by An and Bellare [1]. Our proof technique, applicable to a very general type of construction, turns out to be insightful for constructing (VIL- and) AIL-MACs from FIL-MACs. We propose concrete, essentially optimal AIL-MAC constructions for practical use, Chain-Shift (CS) and Chain-Rotate (CR) (see Fig. 1 and Fig. 2), and prove their security. They use a single key, have constant waste, allow for on-line and parallel processing of the messages, and their security reduction is essentially tight.

The only previously known (practical) VIL-MAC construction, the NI-construction (see Fig. 3), uses two keys, has an upper bound of 2^b on the message length, and is not optimal in terms of the number of applications to the FIL-MAC (especially not for short messages). In Sect. 4.2 we give a simple security proof (using our proof technique) and several improvements of the NI-construction.

2 Preliminaries

2.1 Notation and Definitions

Let $\{0,1\}^L$ denote the set of all bit strings of length L, $\{0,1\}^{\leq N}$ the set of all bit strings of length at most N, $\{0,1\}^*$ the set of all bit strings, and $[n] := \{1,\ldots,n\}$ (with $[0] := \emptyset$). If M is a set, $\#M$ denotes its cardinality. For x, $y \in \{0,1\}^*$, let $|x|$ denote the length of x (in bits), $x\|y$ the concatenation of x and y, $\langle n\rangle_b$ a b-bit encoding of a positive integer $n \leq 2^b$, $x[i]$ the i^{th} bit of x, and $x[i,j] := x[i]\|x[i+1]\|\cdots\|x[j]$ for $1 \leq i < j \leq |x|$. Furthermore, let $\mathrm{RR}(\cdot)$ denote the operator on bit strings that rotates the input by one position to the right, i.e., $\mathrm{RR}(x) := x[L]\|x[1,L-1]$. For a sequence S of elements, $|S|$ denotes its length, and S_i the sequence of the first $i \leq |S|$ elements of S. An encoding $\sigma : \{0,1\}^* \to \{0,1\}^*$ is called *prefix-free* if there are no three strings x, x', $y \in \{0,1\}^*$ such that $x \neq x'$, $|y| \geq 1$, and $\sigma(x)\|y = \sigma(x')$. A *suffix-free* encoding is an encoding which becomes prefix-free if the bit-order of the output is reversed. A *non-trivial* collision for a function f is a pair $x \neq x'$ of inputs for which $f(x) = f(x')$. If $\mathcal{E}$ denotes an event, $\bar{\mathcal{E}}$ denotes the complementary event.

2.2 Security Definition for MACs

A forger F for a MAC $H := \{h_k : \mathcal{M} \to \mathcal{T}\}_{k\in\mathcal{K}}$ has oracle access to $h_k(\cdot)$ (for which k is chosen uniformly at random from $\mathcal{K}$) and can thus learn the tag values for some adaptively chosen messages $m_1, m_2, \ldots$. It then returns a *forgery* (m,τ), i.e., a message m together with a tag τ. The forger F is considered successful if $h_k(m) = \tau$. The only constraint on m is that it must be *new*, i.e., different from all previous messages $m_1, m_2, \ldots$. We refer to a forger F of this kind as a $(\dot{t}, q, \mu, \varepsilon)$-forger, where $\dot{t}$, q, and μ are upper bounds on the running time, the number of messages (or oracle queries), respectively the total length (in bits) of the oracle queries including the forgery message m, and ε is a lower bound on the success probability. Informally, a MAC is considered secure against *existential forgery* under an *adaptive chosen-message attack*, if there is no $(\dot{t}, q, \mu, \varepsilon)$-forger, even for very high values of $\dot{t}$, q, and μ, and a very small value of ε. A forger for a FIL-MAC will be denoted simply as a $(\dot{t}, q, \varepsilon)$-forger, since the parameter μ is determined by q and the message-input-length L, i.e., $\mu = (q+1)L$.

To prove the security of a MAC based on a FIL-MAC one shows that the existence of a $(\dot{t}, q, \mu, \varepsilon)$-forger F for the MAC implies the existence of a $(\dot{t}', q', \varepsilon')$-forger F' for the FIL-MAC, where $\dot{t}'$, q', and ε' are functions of $\dot{t}$, q, μ, and ε. In all our security proofs F is called only once by F'. Therefore, the running time of F' is essentially that of F, i.e., $\dot{t}' \approx \dot{t}$, with some small overhead that is obvious from the construction of F'. We will therefore not bother to explicitly compute the running time of forgers, as this complicates the analysis unnecessarily without providing more insight. Therefore we drop the time parameter $\dot{t}$ in the sequel.

3 Single-key AIL-MACs Based on any FIL-MAC

3.1 FIL-MAC Forgers Based on an AIL-MAC Forger (Single Key)

Let F be a (q, μ, ε)-forger for a MAC C^G, i.e., if k is chosen uniformly at random from $\{0,1\}^\kappa$, and F is allowed at most q oracle queries to C^{g_k} of total length at most μ (including the length of the forgery message), then F returns a valid forgery (m, τ) with probability at least ε. We refer to $F \circ \mathrm{C}^{g_k}$ as the process in which F's queries to C^{g_k} are computed and returned to F, and where F's forgery (m, τ) is verified by computing $\mathrm{C}^{g_k}(m)$. Let us consider the random variables occurring at the interface to g_k (in the process $F \circ \mathrm{C}^{g_k}$). Let z_i denote the i^{th} input to g_k and let $y_i := g_k(z_i)$. The sequences $\mathbf{Z} := (z_1, z_2, \ldots)$ and $\mathbf{Y} := (y_1, y_2, \ldots)$ are thus naturally defined. Note that as soon as the key k and the random coins of F are fixed, all values in $\mathbf{Z}$ and $\mathbf{Y}$ are determined, and also whether F is successful or not. Let $\mathcal{E}$ denote the event that F is successful. Without loss of generality we assume that F's forgery message m is distinct from F's oracle queries. Thus $\mathcal{E}$ occurs if and only if $\mathrm{C}^{g_k}(m) = \tau$.

A FIL-MAC forger F' for G simulates $F \circ \mathrm{C}^{g_k}$ with the help of F and its oracle access to g_k. At some query z_i to g_k it stops the simulation and returns a forgery (z', τ') for g_k (without making any other oracle queries to g_k). Such a forger is characterized by the moment it stops (i.e., i) and the way it produces its forgery. We refer to this as the *strategy* s of F' and let F'_s denote the corresponding forger.

The most simple strategy is the *naïve* strategy s_{na}. $F'_{s_{\mathrm{na}}}$ stops the simulation of $F \circ \mathrm{C}^{g_k}$ at the very last query $\mathbf{z}$ to g_k (i.e., $\mathbf{z}$ is the last entry in $\mathbf{Z}$). Then it returns $(\mathbf{z}, \tau)$ as a forgery, where τ is the forgery tag of F's forgery (m, τ) for C^{g_k}. $F'_{s_{\mathrm{na}}}$ is successful if the following two conditions hold. First, $\mathcal{E}$ occurs, i.e., $\mathrm{C}^{g_k}(m) = \tau$ (and thus $g_k(\mathbf{z}) = \tau$ by definition of $\mathrm{C}^{\cdot}$), and second $\mathbf{z}$ is new, i.e., $\mathbf{z}$ is only the last entry in $\mathbf{Z}$. Let $\mathcal{E}_{\mathrm{new}}$ denote the event that $\mathbf{z}$ is new. Thus $F'_{s_{\mathrm{na}}}$ is successful whenever $\mathcal{E} \wedge \mathcal{E}_{\mathrm{new}}$ occurs.

Imagine that there is a set $\mathcal{S}$ of strategies, such that whenever $\bar{\mathcal{E}}_{\mathrm{new}}$ occurs there exists at least one strategy $s \in \mathcal{S}$ for which F'_s is successful. We refer to such a set $\mathcal{S}$ as *complete* for the construction. Obviously, the set $\mathcal{S} \cup \{s_{\mathrm{na}}\}$ has the property that whenever $\mathcal{E}$ occurs, there is at least one strategy $s \in \mathcal{S} \cup \{s_{\mathrm{na}}\}$ for which F'_s is successful. Thus an overall strategy of F' is to pick its strategy uniformly at random from $\mathcal{S} \cup \{s_{\mathrm{na}}\}$. Its success probability is at least the probability that $\mathcal{E}$ occurs divided by $\#\mathcal{S} + 1$, since the choice of strategy is independent of $\mathcal{E}$. As F''s number of oracle queries is $|\mathbf{Z}|$, which is a random variable, it is convenient to introduce the following function.

Definition 1. *The* expansion *function e of a construction $\mathrm{C}^{\cdot}$ is defined as*

$$e(\tilde{q}, \tilde{\mu}) := \max \left\{ \sum_{i=1}^{\tilde{q}} t(n_i) : n_1, \ldots, n_{\tilde{q}} \in \mathbb{N}_0,\ n_1 + \cdots + n_{\tilde{q}} \leq \tilde{\mu} \right\},$$

where $t(\cdot)$ is the application function of $\mathrm{C}^{\cdot}$.

It follows that $|\mathbf{Z}| \leq e(q+1, \mu)$, since there are at most $q+1$ queries of total length at most μ to C^{g_k} in $F \circ \mathrm{C}^{g_k}$. In general $\#\mathcal{S}$ is a function of $e(q+1, \mu)$.

Proposition 1. *The existence of a complete set $\mathcal{S}$ for a construction $\mathrm{C}^{\cdot}$ and a (q,μ,ε)-forger F for C^G implies the existence of a (q',ε')-forger F' for G, where $q' = e(q+1,\mu)$ and $\varepsilon' = \frac{\varepsilon}{\#\mathcal{S}+1}$.*

Proof. F' picks its strategy s uniformly at random from $\mathcal{S}\cup\{s_{\mathrm{na}}\}$. Let $\mathcal{E}'$ denote the event that F' is successful, and let $\mathcal{E}$ and $\mathcal{E}_{\mathrm{new}}$ be defined as above.

$$\underbrace{\Pr\left[\mathcal{E}'\right]}_{=:\varepsilon'} \geq \underbrace{\Pr\left[\mathcal{E}' \mid \mathcal{E}\wedge\mathcal{E}_{\mathrm{new}}\right]}_{\geq 1/(\#S+1)} \cdot \Pr\left[\mathcal{E}\wedge\mathcal{E}_{\mathrm{new}}\right] + \underbrace{\Pr\left[\mathcal{E}' \mid \bar{\mathcal{E}}_{\mathrm{new}}\right]}_{\geq 1/(\#S+1)} \cdot \underbrace{\Pr\left[\bar{\mathcal{E}}_{\mathrm{new}}\right]}_{\geq \Pr\left[\mathcal{E}\wedge\bar{\mathcal{E}}_{\mathrm{new}}\right]} \geq \underbrace{\frac{\Pr\left[\mathcal{E}\right]}{\#\mathcal{S}+1}}_{=\varepsilon/(\#\mathcal{S}+1)}$$

□

3.2 Deterministic Strategies

An important class of strategies for F' are the deterministic strategies. A deterministic strategy s is characterized by a pair (i,f), where $i \in [e(q+1,\mu)]$ is an index and f a function mapping $(\mathbf{Z}_i, \mathbf{Y}_{i-1})$ to some value $\hat{y}_i \in \{0,1\}^\ell$ (which can be seen as a prediction of y_i). To be more precise, the corresponding forger F'_s stops (the simulation of $F \circ \mathrm{C}^{g_k}$) at query z_i and returns $(z_i, \hat{y}_i)$ as a forgery.[1] The forger is successful if $\hat{y}_i = y_i$ and if z_i is new, i.e., not contained in the sequence $\mathbf{Z}_{i-1}$. Next follow three particular sets of strategies, which will be used in the sequel:

- Let $s_{i,y}$ (with $y \in \{0,1\}^\ell$) denote the strategy of stopping at query z_i and returning (z_i, y) as a forgery. Note that whenever the event occurs that g_k outputs y, i.e., when y is an entry in $\mathbf{Y}$, then there is at least one strategy $s \in \mathcal{S}_y := \{s_{i,y} | i \in [e(q+1,\mu)]\}$ for which F'_s is successful. We have
$$\#\mathcal{S}_y = e(q+1,\mu). \tag{1}$$
- Let $s_{\mathrm{coll},i,j}$ (with $i > j$) denote the strategy of stopping at query z_i and returning (z_i, y_j) as a forgery. Note that whenever a non-trivial collision for g_k occurs, i.e., $\alpha, \beta \in [|\mathbf{Z}|]$ satisfying $z_\alpha \neq z_\beta$ and $y_\alpha = y_\beta$, then there is at least one strategy $s \in \mathcal{S}_{\mathrm{coll}} := \{s_{\mathrm{coll},i,j} | i,j \in [e(q+1,\mu)], i > j)\}$ for which F'_s is successful. The cardinality of $\mathcal{S}_{\mathrm{coll}}$ is
$$\#\mathcal{S}_{\mathrm{coll}} = e(q+1,\mu)^2/2 - e(q+1,\mu)/2. \tag{2}$$
- Let $s_{\mathrm{coll2},i,j,a,\mathrm{left}}$ (with $a \in \{0,1\}$ and $i > j$) denote the strategy of stopping at input z_i and returning $(z_i, a\|y_j[1,\ell-1])$ as a forgery, and let $s_{\mathrm{coll2},i,j,a,\mathrm{right}}$ denote the strategy of stopping at input z_i and returning $(z_i, y_j[2,\ell]\|a)$ as a forgery. Note that whenever the event occurs that there are $\alpha, \beta \in [|\mathbf{Z}|]$ satisfying $z_\alpha \neq z_\beta$ and $g_k(z_\alpha)[2,\ell] = g_k(z_\beta)[1,\ell-1]$, then there is a strategy $s \in \mathcal{S}_{\mathrm{coll2}} := \{s_{\mathrm{coll2},i,j,a,d} | i,j \in [e(q+1,\mu)], i > j, a \in \{0,1\}, d \in \{\mathrm{left},\mathrm{right}\}\}$ for which F'_s is successful. The cardinality of $\mathcal{S}_{\mathrm{coll2}}$ is
$$\#\mathcal{S}_{\mathrm{coll2}} = 2 \cdot e(q+1,\mu)^2 - 2 \cdot e(q+1,\mu). \tag{3}$$

[1] If $i > |\mathbf{Z}|$ the forger aborts.

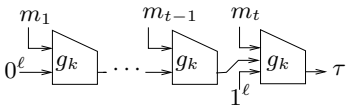

Fig. 1. The Chain-Shift (CS) construction

3.3 The Chain-Shift (CS) Construction

The CS-construction uses any FIL-MAC $G := \{g_k : \{0,1\}^{b+\ell} \to \{0,1\}^\ell\}_{k\in\{0,1\}^\kappa}$ with compression $b \geq \ell$, to construct an AIL-MAC $\mathrm{CS}^G := \{\mathrm{CS}^{g_k} : \{0,1\}^* \to \{0,1\}^\ell\}_{k\in\{0,1\}^\kappa}$. For a message $m \in \{0,1\}^*$ of length $n := |m|$, the tag $\tau = \mathrm{CS}^{g_k}(m)$ is computed according to the following recursion (as depicted in Fig. 1). Parse m into a sequence of b-bit blocks $m_1, \ldots, m_{t-1}$ and a $(b-\ell)$-bit block m_t, such that $m_1 \| \cdots \| m_t = m \| 10^\nu$ for some $\nu \in \{0, \ldots, b-1\}$:

$$y_0 := 0^\ell,\ y_i := g_k\left(y_{i-1} \| m_i\right) \ \text{ for } \ i \in [t-1], \ \text{ and } \ \tau := g_k(1^\ell \| y_{t-1} \| m_t).$$

The waste $w(n) = t(n) \cdot b + \ell - n = \lceil (n+1+\ell)/b \rceil \cdot b + \ell - n \leq L + \ell$ is upper bounded by a constant, and on-line processing is possible.

Theorem 1.[2] *A (q, μ, ε)-forger F for CS^G implies a (q', ε')-forger F' for G, where $q' = \left\lfloor \frac{\mu}{b} \right\rfloor + 2(q+1)$ and $\varepsilon' = \frac{\varepsilon}{q'^2/2 + 3q'/2 + 1}$.*

Proof. We apply Proposition 1 and show that $\mathcal{S} := \mathcal{S}_{\mathrm{coll}} \cup \mathcal{S}_{0^\ell} \cup \mathcal{S}_{1^\ell}$ is complete for $\mathrm{CS}^{\cdot}$ by proving that whenever the last entry $\mathbf{z}$ in $\mathbf{Z}$ is not new, then there is a non-trivial collision in g_k, or an output from g_k that equals 0^ℓ or 1^ℓ.

Assume that $\mathbf{z}$ is not new. Let $\tilde{z}_1, \ldots, \tilde{z}_t$ denote the sequence of queries to g_k resulting from the last query m_β to CS^{g_k}. Note that m_β is the forgery message of F and thus new. Since $\tilde{z}_t = \mathbf{z}$ is not new, $\tilde{z}_t$ must have been an earlier query to g_k, resulting from some query m_α (with $\alpha \leq \beta$) to CS^{g_k}. Let $\tilde{z}'_1, \ldots, \tilde{z}'_{t'}$ denote the sequence of queries to g_k in the computation of $\mathrm{CS}^{g_k}(m_\alpha)$. There are three cases to distinguish, depending on the index $i \in [t']$ for which $\tilde{z}_t = \tilde{z}'_i$.

At the end of the chain ($\tilde{z}_t = \tilde{z}'_{t'}$): First, we note that this can not be the case if $\alpha = \beta$, since in that case $\tilde{z}'_{t'}$ is not an earlier occurring query. Thus we have (the non-trivial collision) $m_\alpha \neq m_\beta$ satisfying $\mathrm{CS}^{g_k}(m_\alpha) = \mathrm{CS}^{g_k}(m_\beta)$. Without loss of generality assume that $t' \geq t$. Now, either there exist an index $j \in [t-1]$ such that $\tilde{z}_{t-j} \neq \tilde{z}'_{t'-j}$ and $\tilde{z}_{t-j+1} = \tilde{z}'_{t'-j+1}$, i.e., a non-trivial collision in g_k occurs (since $\tilde{z}_{t-j+1} = \tilde{z}'_{t'-j+1}$ implies $g_k(\tilde{z}_{t-j}) = g_k(\tilde{z}'_{t'-j})$) or $\tilde{z}'_{t'-t+1} = \tilde{z}_1 = 0^\ell \| v$ for some $v \in \{0,1\}^b$, which implies $g_k(\tilde{z}'_{t'-t}) = 0^\ell$ (with $t' - t \geq 1$ since $m_\alpha \neq m_\beta$).

[2] An and Bellare point out in [1] that the security loss of roughly $(\mu/b)^2$ is unavoidable for iterative constructions of this nature. It is shown using birthday attacks illustrated by Preneel and Van Oorschot [11].

In the middle of the chain ($\tilde{z}_t = \tilde{z}'_i$ with $1 < i < t'$): We have $1^\ell \| v = \tilde{z}_t = \tilde{z}'_i = g_k(\tilde{z}'_{i-1}) \| v$, for some $v \in \{0,1\}^b$. Thus g_k outputs 1^ℓ.

At the beginning of the chain ($\tilde{z}_t = \tilde{z}'_1$ and $t' > 1$): This case is obviously impossible, since $\tilde{z}_t = 1^\ell \| v \neq 0^\ell \| v' = \tilde{z}'_1$ for any $v, v' \in \{0,1\}^b$.

By definition of $e(q+1,\mu)$, there is a sequence $n_1, \ldots, n_{q+1} \in \mathbb{N}_0$ such that:

$$e(q+1,\mu) = \sum_{i=1}^{q+1} t(n_i) \leq \left\lfloor \frac{\mu + (q+1)L}{b} \right\rfloor \leq \left\lfloor \frac{\mu}{b} + 2(q+1) \right\rfloor =: q'.$$

Thus $\#\mathcal{S} + 1 \leq (q'^2/2 - q'/2) + q' + q' + 1 \leq q'^2/2 + 3q'/2 + 1$ by (1) and (2). $\square$

Improving the Waste for Short Messages. We improve the efficiency of the CS-construction for $n := |m| < rb$, where $r \in \mathbb{N}_0$ is a design parameter. This is relevant (see for example [9]). The computation of the tag τ is redefined for messages m of length shorter than rb as follows. Parse m into a sequence of b-bit blocks $m_1, \ldots, m_t$ such that $m_1 \| \ldots \| m_t = m \| 10^\nu$ where $\nu \in \{0, \ldots, b-1\}$:

$$y_0 := \langle t \rangle_\ell,\ y_i := g_k(y_{i-1} \| m_i) \ \text{ for } \ i \in [t], \ \text{ and } \ \tau := y_t.$$

Now, $t(n) = \lceil (n+1)/b \rceil$ if $n < rb$ (and $t(n) = \lceil (n+1+\ell)/b \rceil$ if $n \geq rb$). The proof that $\mathcal{S}_{\text{coll}} \cup \mathcal{S}_{0^\ell} \cup \mathcal{S}_{1^\ell} \cup (\cup_{i=1}^{r} \mathcal{S}_{\langle i \rangle_\ell})$ is complete for the construction is omitted. The only modification of Theorem 1 is thus that $\epsilon' = \frac{\epsilon}{q'^2/2 + (3/2+r)q' + 1}$, i.e., the reduction is essentially as tight (as for $r = 0$) for reasonable r's.

Parallelizing the CS-Construction. We modify the CS-construction to allow $c \geq 1$ processors to compute the tag in parallel, achieving a speed up by a factor of roughly c for long messages. The tag τ of an n-bit message m is computed according to the following recursion:

1. If $c \leq \lceil (n+1)/b \rceil$ then set $c' := c$, and else set $c' := \lceil (n+1)/b \rceil$.
2. Parse m into $m_1 \| \cdots \| m_{c't} = m \| 10^\nu$, where $m_1, \ldots, m_{c't}$ are b-bit blocks and $\nu \in \{0, \ldots, c'b - 1\}$. Set $m_{i,j} := m_{i+(j-1)c'}$ for $i \in [c']$ and $j \in [t]$.
3. Set $y_{i,0} := 0^\ell$, and compute $y_{i,j} := g_k(y_{i,j-1} \| m_{i,j})$ for $i \in [c']$ and $j \in [t]$.
4. Return $\tau := \text{CS}^{g_k}(y_{1,t} \| \cdots \| y_{c',t})$.[3]

The waste remains constant and the on-line property is preserved. We omit the proof that $\mathcal{S} = \mathcal{S}_{\text{coll}} \cup \mathcal{S}_{0^\ell} \cup \mathcal{S}_{1^\ell}$ is complete for the construction, as it is similar to the proof that $\mathcal{S}$ is complete for the CS-construction.

[3] The construction can be further parallelized by replacing step 4 as follows. For simplicity assume $b = \ell$ (the generalization to $b \geq \ell$ is straight forward). Apply g_k to every pair of adjacent blocks in $(y_{1,t}, \cdots, y_{c',t})$, resulting in a new sequence of $\lceil c'/2 \rceil$ blocks, and repeat this until a single block y is obtained. Then set $\tau := g_k(1^\ell \| y)$.

By setting $c := \infty$ this construction is *fully* parallelized (FP) (here meaning that the computation time is in $\Theta(\log(n))$ when arbitrary many processors are available) with $w(n) \in \Theta(n)$. From a theoretical viewpoint it would be interesting to see whether FP single-key AIL-MAC constructions with $w(n) \in \Theta(1)$ exists. There are FP single-key AIL-MAC constructions with $w(n) \in \Theta(\log(n))$ and FP 2-key AIL-MAC constructions with $w(n) \in \Theta(1)$.

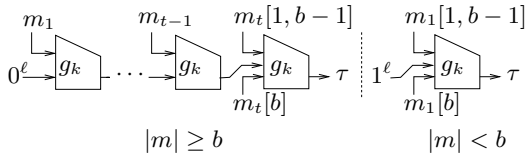

Fig. 2. The Chain-Rotate (CR) construction

3.4 The Chain-Rotate (CR) Construction

The purpose of presenting this single-key AIL-MAC construction is twofold. First, it shows that constant waste and on-line processing is possible (even) with compression $b < \ell$. Second, it demonstrates the generality of our proof technique.

The CR-construction transforms any FIL-MAC $G := \{g_k : \{0,1\}^{b+\ell} \to \{0,1\}^\ell\}_{k\in\{0,1\}^\kappa}$ into an AIL-MAC $\mathrm{CS}^G := \{\mathrm{CS}^{g_k} : \{0,1\}^* \to \{0,1\}^\ell\}_{k\in\{0,1\}^\kappa}$. The tag $\tau = \mathrm{CR}^{g_k}(m)$ of an n-bit message m is computed as follows (see Fig. 2). Parse m into a sequence of b-bit blocks $m_1, \ldots, m_t$ such that $m_1 \| \cdots \| m_t = m \| 10^\nu$ with $\nu \in \{0, \ldots, b-1\}$. If $t > 1$, set $y_0 := 0^\ell$ and else set $y_0 := 1^\ell$:

$$y_i := g_k(y_{i-1} \| m_i) \text{ for } i \in [t-1] \text{ and } \tau := g_k(\mathrm{RR}(y_{t-1} \| m_t)).$$

The waste is $w(n) = \lceil (n+1)/b \rceil \cdot b + \ell - n \leq L$, and on-line processing is possible.

Theorem 2. *A (q, μ, ε)-forger F for CR^G implies a (q', ε')-forger F' for G, where $q' = \lfloor \frac{\mu}{b} \rfloor + q + 1$ and $\varepsilon' = \frac{\varepsilon}{5q'^2/2 + 3q'/2 + 1}$.*

Proof (Sketch). We apply Proposition 1. With a case study similar to that for the $\mathrm{CS}^{\cdot}$ (here omitted), one shows that $\mathcal{S} := \mathcal{S}_{\mathrm{coll}} \cup \mathcal{S}_{\mathrm{coll2}} \cup \mathcal{S}_{0^\ell} \cup \mathcal{S}_{1^\ell} \cup \mathcal{S}_{0^{\ell-1}1} \cup \mathcal{S}_{01^{\ell-1}}$ is complete for $\mathrm{CR}^{\cdot}$. There exist $n_1, \ldots, n_{q+1} \in \mathbb{N}_0$ such that:

$$e(q+1, \mu) = \sum_{i=1}^{q+1} t(n_i) \leq \sum_{i}^{q+1} \left\lceil \frac{n_i + 1}{b} \right\rceil \leq \left\lfloor \sum_{i=1}^{q+1} \frac{n_i + b}{b} \right\rfloor \leq \left\lfloor \frac{\mu}{b} \right\rfloor + q + 1 =: q'.$$

As a consequence, $\#\mathcal{S} + 1 \leq 5q'^2/2 + 3q'/2 + 1$ by (1), (2), and (3). □

Parallelizing the CR-Construction. The CR-construction can be parallelized in a similar way as the CS-construction. Just replace CS by CR in step 4 of the corresponding paragraph of Sect. 3.3. As for the CR-construction the set $\mathcal{S} := \mathcal{S}_{\mathrm{coll}} \cup \mathcal{S}_{\mathrm{coll2}} \cup \mathcal{S}_{0^\ell} \cup \mathcal{S}_{1^\ell} \cup \mathcal{S}_{0^{\ell-1}1} \cup \mathcal{S}_{01^{\ell-1}}$ is complete for the construction.

4 Comparison with the NI-Construction and Variations

The security analysis described in Sect. 3.1 can be generalized to multiple-key constructions. Motivated by the NI-construction (see Fig. 3), we consider constructions $\mathrm{C}^{\cdot,\cdot}$ (using two instantiations g_{k_1} and g_{k_2} of G to construct an instantiation $\mathrm{C}^{g_{k_1}, g_{k_2}}$ of the MAC $\mathrm{C}^{G,G}$), where one of the instantiations (say g_{k_2} without loss of generality) is invoked at the end of the computation. We prove the security of the NI-construction and give several improvements thereof.

4.1 FIL-MAC Forgers Based on an AIL-MAC Forger (2 Keys)

Let F denote a (q, μ, ε)-forger for the MAC $\mathrm{C}^{G,G}$. As before let $F \circ \mathrm{C}^{g_{k_1}, g_{k_2}}$ denote the process in which for each query $\tilde{m}$ issued by F, the corresponding tag $\mathrm{C}^{g_{k_1}, g_{k_2}}(\tilde{m})$ is computed and returned to F, and once F returns a forgery (m, τ), the forgery is verified by computing $\mathrm{C}^{g_{k_1}, g_{k_2}}(m)$. Let $\mathbf{Z}^i := (z_1^i, z_2^i, \ldots)$ and $\mathbf{Y}^i := (y_1^i, y_2^i, \ldots)$ be the sequence of inputs respectively outputs occurring at the interface to instantiation g_{k_i} (for $i \in \{1, 2\}$).

The FIL-MAC forger F' simulates $F \circ \mathrm{C}^{g_{k_1}, g_{k_2}}$ by letting its own oracle simulate one of the instantiations g_{k_i} (say the instantiation *under attack*) and by choosing a random key for the other, but stops the simulation at some query z_j^i to its oracle and returns a forgery (without making any further query to any FIL-MAC instantiation). This is equivalent to first instantiating g_{k_1} and g_{k_2} (by choosing the keys k_1, k_2 uniformly at random) and then letting F' specify which instantiation to attack, i.e., consider as its own oracle, after which the key to the other instantiation is revealed to F'. We adopt this view. Any such forger is characterized by its *strategy*, i.e., which instantiation it attacks (i.e., i), the moment it stops (i.e., j), and the way it produces its forgery.

Let s_{na} denote the naïve strategy described in Sect. 3.1, with the only modification that the second instantiation, g_{k_2} is put under attack (recall that the tag τ is an output of g_{k_2}). $F'_{s_{\mathrm{na}}}$ stops at the very last query $\mathbf{z}$ to g_{k_2} and returns $(\mathbf{z}, \tau)$ as a forgery. Of course F' is successful if the following two conditions hold. First, $\mathcal{E}$ occurs, i.e., $\mathrm{C}^{g_{k_1}, g_{k_2}}(m) = \tau$ (and thus $g_{k_2}(\mathbf{z}) = \tau$),[4] and second $\mathcal{E}_{new}$ holds, i.e., $\mathbf{z}$ is new for g_{k_2} or equivalently $\mathbf{z}$ is only the last entry in $\mathbf{Z}^2$.

Imagine as before, that a *complete* set of strategies $\mathcal{S}$ exists, i.e., a set for which whenever $\bar{\mathcal{E}}_{\mathrm{new}}$ occurs, there exists a strategy $s \in \mathcal{S}$ for which F'_s is successful. Then an overall strategy of F' is to pick its strategy uniformly at random from $\mathcal{S} \cup \{s_{\mathrm{na}}\}$. Its success probability is at least the probability that $\mathcal{E}$ occurs (i.e., ε) divided by the number $\#\mathcal{S} + 1$ of strategies, since the choice of strategy is independent of the event $\mathcal{E}$. Since F''s number of queries to its oracle is upper bounded by $\max\{|\mathbf{Z}^1|, |\mathbf{Z}^2|\}$, which is a random variable, it is convenient to introduce the expansion function for each instantiation, i.e., for $i \in \{1, 2\}$, let $e^i(\tilde{q}, \tilde{\mu}) := \max\{\sum_{j=1}^{\tilde{q}} t^i(n_j) : n_1, \ldots, n_{\tilde{q}} \in \mathbb{N}_0, n_1 + \cdots + n_{\tilde{q}} \leq \tilde{\mu}\}$. Thus $|\mathbf{Z}^i| \leq e^i(q + 1, \mu)$. Proposition 1 generalizes as follows.

Proposition 2. *The existence of a complete set $\mathcal{S}$ for a construction $\mathrm{C}^{\cdot,\cdot}$ and a (q, μ, ε)-forger F for $\mathrm{C}^{G,G}$ implies a (q', ε')-forger F' for G, where $q' = \max(e^1(q + 1, \mu), e^2(q + 1, \mu))$ and $\varepsilon' = \frac{\varepsilon}{\#\mathcal{S}+1}$.*

A *deterministic* strategy s is now characterized by a triple of values (i, j, f), where i denotes the instantiation to attack, z_j^i the moment to stop, and f a function mapping $(\mathbf{Z}_j^i, \mathbf{Y}_{j-1}^i)$ to some value $\hat{y}_j^i \in \{0, 1\}^\ell$. The pair $(z_j^i, \hat{y}_j^i)$ is the forgery of F'_s. The sets of deterministic strategies introduced in Sect. 3.2 is naturally defined for each instantiation. Let $\mathcal{S}_y^i$, $\mathcal{S}_{\mathrm{coll}}^i$, and $\mathcal{S}_{\mathrm{coll2}}^i$ denote the corresponding sets for the i^{th} instantiation.

[4] We assume w.l.o.g. that F's forgery message m is distinct from its oracle queries.

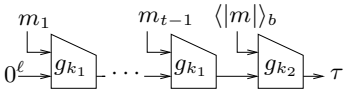

Fig. 3. The nested iterated (NI) construction

4.2 The NI-Construction

The NI-construction [1] transforms any FIL-MAC G into a VIL-MAC $\mathrm{NI}^{G,G} := \{\mathrm{NI}^{g_{k_1},g_{k_2}} : \{0,1\}^{\leq 2^b} \to \{0,1\}^\ell\}_{k_1,k_2 \in \{0,1\}^\kappa}$. For a message $m \in \{0,1\}^{\leq 2^b}$ of length $n := |m|$, the tag $\tau = \mathrm{NI}^{g_{k_1},g_{k_2}}(m)$ is computed according to the following recursion (as illustrated in Fig. 3). Break m into $t-1 = \lceil n/b \rceil$ blocks $\{m_i\}_{i=1}^{t-1}$ of length b, where m_{t-1} is padded with zeroes if necessary, and set $m_t := \langle n \rangle_b$:

$$y_0 := 0^\ell,\ y_i := g_{k_1}(y_{i-1} \| m_i) \ \text{ for } \ i \in [t-1], \ \text{ and } \ \tau := g_{k_2}(y_{t-1} \| m_t). \qquad (4)$$

The waste is $w(n) = t(n) \cdot b + \ell - n = \lceil \frac{n}{b} + 1 \rceil \cdot b + \ell - n \leq L + b$, and on-line processing is possible. Note that the message space is VIL, due to $m_t := \langle n \rangle_b$.

Theorem 3. *A (q, μ, ε)-forger F for $\mathrm{NI}^{G,G}$ implies a (q', ε')-forger F' for G, where $q' = \lfloor \frac{\mu}{b} \rfloor + q + 1$ and $\varepsilon' = \frac{\varepsilon}{q'^2/2 - q'/2 + 1}$.*

Proof. We show that $\mathcal{S}^1_{\mathrm{coll}}$ is complete for $\mathrm{NI}^{\cdot,\cdot}$ by proving that whenever the last entry $\mathbf{z}$ in $\mathbf{Z}^2$ is not new, then a non-trivial collision in g_{k_1} occurs. Let m_β denote the forgery message of F. Since $\mathbf{z}$ is not new, there is a query m_α (issued by F and different from m_β) with same input to g_{k_2}. Thus we have $|m_\alpha| = |m_\beta|$, and (the non-trivial collision) $m_\alpha \neq m_\beta$ satisfying $\mathrm{NI}^{g_{k_1},g_{k_2}}(m_\alpha) = \mathrm{NI}^{g_{k_1},g_{k_2}}(m_\beta)$. Since $m_\alpha \neq m_\beta$, all corresponding intermediate values in the computation chains can not be the same. As a consequence a non-trivial collision in g_{k_1} occurs.

By definition of $e^1(q+1, \mu)$, there is a sequence $n_1, \ldots, n_{q+1} \in \mathbb{N}_0$ such that:

$$e^2(q+1,\mu) \leq e^1(q+1,\mu) = \sum_{i=1}^{q+1} t^1(n_i) \leq \left\lfloor \sum_{i=1}^{q+1} \frac{n_i + b - 1}{b} \right\rfloor \leq \left\lfloor \frac{\mu}{b} + q + 1 \right\rfloor =: q'.$$

Thus $\#\mathcal{S}^1_{\mathrm{coll}} + 1 \leq q'^2/2 - q'/2 + 1$ by (2). Proposition 2 concludes the proof. □

Improvements on the NI-Construction

1. By replacing $y_0 := 0^\ell$ with a message block, the waste decreases by ℓ bits, the security reduction is slightly tighter, and the on-line property is of course preserved. The security proof is identical to that of the NI-construction.
2. The block $m_t := \langle n \rangle_b$, encoding the message length, is superfluous. It can be replaced by a message block with appropriate padding. This decreases the waste of the construction, improves the tightness of the reduction, lifts the message space to AIL, and preserves the on-line property. To be more precise the message m is parsed into a sequence of b-bit blocks $m_1, \ldots, m_t$ such that $m_1 \| \ldots \| m_t = m \| 10^\nu$ with $\nu \in \{0, \ldots, b-1\}$ and processed according to (4). It is straight forward to see that $\mathcal{S}^1_{\mathrm{coll}} \cup \mathcal{S}^1_{0^\ell}$ is complete for the construction.

3. If the block encoding the message length is used as the first block instead of the last or if any other prefix-free encoding of the message into blocks is used, the two keys can actually be replaced by a single key. By choosing an appropriate prefix-free encoding (for example the one on page 126 in [6]) the message space can be lifted to AIL, at the cost of having $w(n) \in \Theta(\log(n))$. We conjecture that linear waste, i.e., $w(n) \in \Theta(n)$ is needed for the on-line property. It is easy to verify that $\mathcal{S}_{\text{coll}} \cup \mathcal{S}_{0^\ell}$ is complete for the construction.

5 Conclusions

A general paradigm for constructing VIL- and AIL-MACs from any FIL-MAC was presented. The design goals were minimal key-length, optimal waste, as well as suitability for on-line and parallel processing of the messages. Our single-key AIL-MAC constructions, CS and CR, have constant waste, allow for on-line and parallel processing of the message, and have essentially tight security reductions.

References

1. J. H. An and M. Bellare. Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. In *Advances of Cryptology — CRYPTO '99*, volume 1666 of *LNCS*, pages 252–269. Springer-Verlag, 1999.
2. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances of Cryptology — CRYPTO '96*, volume 1109 of *LNCS*, pages 1–15. Springer-Verlag, 1996.
3. M. Bellare, J. Guérin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Advances of Cryptology — CRYPTO '95*, volume 963 of *LNCS*, pages 15–28. Springer-Verlag, 1995.
4. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In *Journal of Computer and System Sciences*, 61(3):362–399, 2000.
5. I. Damgård. A design principle for hash functions. In *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1990.
6. U. Maurer. Indistinguishability of random systems. In *Advances of Cryptology — EUROCRYPT '02*, volume 2332 of *LNCS*, pages 110–132. Springer-Verlag, 2002.
7. R. Merkle. A certified digital signature. In *Advances in Cryptology — CRYPTO '89*, volume 435 of *LNCS*, pages 218–232. Springer-Verlag, 1990.
8. M. Naor and O. Reingold. From unpredictability to indistinguishability: A simple construction of pseudo-random functions from MACs (extended abstract). In *Advances in Cryptology — CRYPTO '98*, volume 1462 of *LNCS*, pages 267–282. Springer-Verlag, 1998.
9. S. Patel. An efficient MAC for short messages. In *Selected Areas in Cryptography*, volume 2595 of *LNCS*, pages 352–368. Springer-Verlag, 2003.
10. E. Petrank and C. Rackoff. CBC MAC for real-time data sources. In *Journal of Cryptology*, 13(3):315–338, 2000.
11. B. Preneel and P. C. van Oorschot, MDx-MAC and building fast MACs from hash functions. In *Advances in Cryptology — CRYPTO '95*, volume 953 of *LNCS*, pages 1–14. Springer-Verlag, 1995.