Weak-Key Analysis of POET

Mohamed Ahmed Abdelraheem, Andrey Bogdanov, and Elmar Tischhauser

Department of Applied Mathematics and Computer Science
Technical University of Denmark, Denmark
{mohab,anbog,ewti}@dtu.dk

Abstract. We evaluate the security of the recently proposed authenticated encryption scheme POET with regard to weak keys when its universal hash functions are instantiated with finite field multiplications. We give explicit constructions for weak key classes not covered by POET's weak key testing strategy, and demonstrate how to leverage them to obtain universal forgeries.

1 Introduction

POET is a recent proposal by Abed *et al.* for an online authenticated encryption scheme, and has also been submitted to the ongoing CAESAR competition [1, 5]. It uses a combination of Rogaway's XEX construction with the AES as underlying block cipher and AXU hash functions to produce the XOR masks. One recommended variant instantiates these hash functions as multiplications with keys in $\mathbb{F}_2^{128}$. In this case, the input to the block ciphers top layer of masks in POET's XEX structure basically consists of a polynomial hash evaluation of the message inputs.

Polynomial hashing is known to have issues with weak key classes [3, 6, 7]. In this paper, we analyze the impact of weak keys on POET.

2 The authenticated online cipher POET

In this section, we briefly describe the POET authenticated online cipher [1].

A schematic description of POET is given in Fig. 1. It uses a combination of Rogaway's XEX construction with a chain of AXU hash function evaluations F_t to update the first (top) layer of masks, the bottom layer of masks being generated by applying another AXU F_b to the previous output of the block cipher calls. Associated data (AD) and the nonce are processed in a PMAC-like fashion to produce a value τ which is then used as the initial chaining value for both top and bottom mask layers, as well as for generating the authentication tag T. Five keys $L, K, L_{\text{top}}, L_{\text{bot}}$ and L_T are derived from a user key as encryptions of the constants $1, \ldots, 5$. K denotes the block cipher key, L is used as the mask in the AD processing, and L_T is used as a mask for computing the tag. The "header" H encompasses the associated data (if present) and includes the nonce in its last block. S denotes the encryption of the bit length of the message M, i.e. $S = E_K(|M|)$. The inputs and outputs of the i-th block cipher call during message processing are denoted by X_i and Y_i, respectively.

In a recommended variant of POET, the functions F_t and F_b are given by $F_t(x) = L_{\text{top}} \cdot x$ and $F_b(x) = L_{\text{bot}} \cdot x$, with the multiplication taken in $\mathbb{F}_2^{128}$. This is also the variant that we consider in this paper. The top AXU hash chain then corresponds to the evaluation of a polynomial hash in $\mathbb{F}_2^{128}$:

$$g_t(X) = \tau {L_{\text{top}}}^m + \sum_{i=1}^{m} X_i {L_{\text{top}}}^{m-i},$$

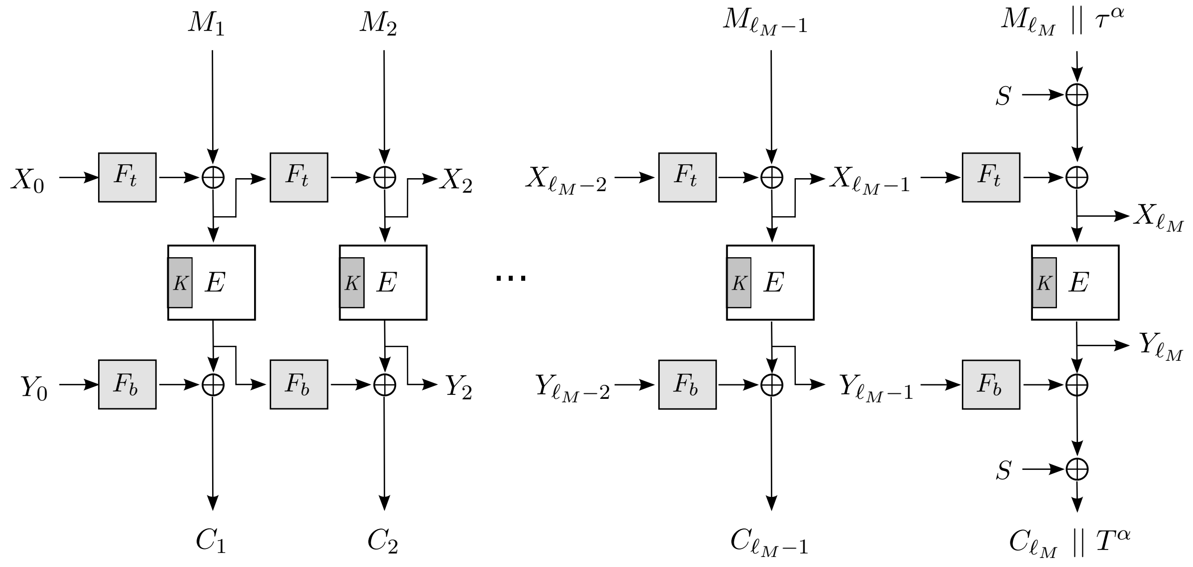

Fig. 1: Schematic description of POET [1]

with g_t being evaluated at $X = M_1, \ldots, M_{m-1}, M_m \oplus S$.

For integral messages (*i.e.*, with a length a multiple of the block size), the authentication tag T then generated as $T = T^\beta$ with empty Z, as shown in Fig. 2. Otherwise, the tag T is the concatenation of the two parts T^α and T^β, see Fig. 1 and 2.

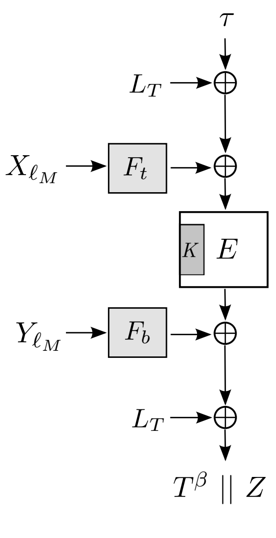

Fig. 2: Second-part tag generation in POET [1]

3 Weak Keys in Polynomial Hashing

We start by first describing polynomial hashing authentication schemes. Then we describe the main observation on polynomial hashing authentication schemes made by Procter and Cid in their FSE paper [6] which enables them to give a general forgery attack on polynomial hashing authentication schemes. As mentioned in [6], most of the previous attacks [2–4,7] on the most well known polynomial hashing scheme, McGrew and Viega's Galois/Counter Mode (GCM), turned out to be a special case of their general forgery attack.

3.1 Polynomial Hashing Authentication Schemes

A polynomial hash-based authentication scheme processes an input consisting of a key H and plaintext/ciphertext $M = (M_1||M_2||\cdots||M_l)$, where each $M_i \in \mathbb{F}_2^{128}$, by evaluating the polynomial

$$h_H(M) = \sum_{i=1}^{t} M_i H^i \in \mathbb{F}_2^{128}.$$

The polynomial $h_H(M)$ is used to construct fast and secure MACs. For instance, the GCM tag generation can be described as follows

$$MAC_{H||k}(M) = E_k(N) \oplus h_H(M),$$

where M is the ciphertext produced using a counter mode block cipher E_k, N is the nonce and $H = E_k(0)$ and k is the secret key. One can see that by repeating the nonce N, one can create forgeries if a hash collision is found on $h_H(M)$. For example, in [7], Saarinen created a forgery on GCM when the hash key H generates a cyclic subgroup of order t, in other words when $H^{t+1} = H$. Hash keys satisfying this property are called weak keys since they allow the attacker to create a valid forgery by simply swapping any two message blocks M_i and M_{i+jt}. Next we describe a general version of Saarinen's cycling attack which we will use throughout the paper.

3.2 Procter and Cid's Forgery Attack

The main observation of [6] can be described as follows. Let H be the unknown hash key. Assume that $q(x) = \sum_{i=1}^{r} q_i x^i$ and that $q(H) = 0$. Assume that $M = (M_1||M_2||\cdots||M_l)$ and that $l < r$. Then

$$h_H(M) = \sum_{i=1}^{r} M_i H^i = \sum_{i=1}^{l} M_i H^i + \sum_{i=1}^{r} q_i H^i = \sum_{i=1}^{r} (M_i + q_i) H^i = h_H(M + Q)$$

where $Q = q_1||\cdots||q_r$. Note that we need to pad M with zeros since $l < r$. Considering the GCM scheme, if we know that (N, M, T) is valid then $(N, M + Q, T)$ is valid if $q(H) = 0$ where $H \in \mathbb{F}_2^{128}$. This gives a forgery probability $p = \frac{\#\text{roots of q(x)}}{2^{128}}$. Therefore, in order to have a forgery using the polynomial $q(x)$ with high probability, $q(x)$ should have a high degree and preferably no repeated roots. Next we describe how to choose a forgery polynomial $q(x)$ with high forgery probability.

3.3 Weak keys and POET's avoidance strategy

The specification of POET with $\mathbb{F}_2^{128}$ multiplications discusses the issue of weak keys and proposes to perform a check on $L, L_{\text{top}}, L_{\text{bot}}$ during the key generation phase [1]. No precise description is given on how this check is performed, the reference to Saarinen's cycling attacks [7] and the suggested weak key probability of 2^{-96}, however, imply that only some basic cycling attacks (swapping of blocks) are excluded. This does not take into account the results of [6], where it is demonstrated that arbitrary polynomials (not limited to two terms) can be used as forgery polynomials. Moreover, in this general setting, *any* key can be considered potentially weak. In any event, the description of POET does not allow for ruling out a class of weak keys with more than 2^{32} elements.

Even if POET's weak key "detection" strategy were modified to only allow generators of the multiplicative group of $\mathbb{F}_2^{128}$, there would still be weak key classes, since any element of $\mathbb{F}_2^{128}$ can be a root of some forgery polynomial. We further note that the order (as a group element) of the weak key is not related to the degree of the forgery polynomial and thus the query length. A high-degree polynomial is only needed to obtain a better success probability with random keys.

3.4 Choosing Forgery Polynomials

Procter and Cid described three methods to construct a forgery polynomial [6]. The trivial construction is to compute $q(x) = \prod_i (x - K_i)$ for as many secret keys K_i as possible in order to gain the desired forgery probability. The second method is to multiply distinct irreducible polynomials in the subfields of $\mathbb{F}_2^{128}$. As mentioned in [6], this method embodies the polynomials used in Saarinen's cycling attack [7]. However, it differs from Saarinen's attack since it contains roots from different subgroups' elements while Saarinen's attack uses polynomials whose roots are in the same subgroup. The third method uses random polynomials in $\mathbb{F}_2^{128}$ which as noted in [6] might not split in $\mathbb{F}_2^{128}$.

In this paper, we use the second method in order to build a forgery polynomial with probability p, $\frac{2^{32}}{2^{128}} < p < \frac{2^{62}}{2^{128}}$.

Forgery Polynomials Suitable for POET Another possibility to construct a forgery polynomial is to consider the subgroups with orders that are not prime. According to [1] the maximum message length in POET is less than 2^{64} blocks. This implies that the order of the subgroup should be less than that as well. There are 240 subgroups with such orders out of the total of 512 subgroups. Furthermore, in order to use keys which is not ruled out by POET's weak key avoidance test, which does not exclude more that 2^{32} keys, we choose subgroups with order more than 2^{32}. There are 163 subgroups out of the total of 512 subgroups that have orders in the interval $(2^{32}, 2^{62})$. Using the polynomial $q(x) = x^{n+1} - x$ where n is the order of the subgroup, we can get a forgery with probability $\frac{n+1}{2^{128}}$. Any of the corresponding polynomials of those 163 subgroups will give us a forgery attack with probability p, $\frac{2^{32}}{2^{128}} < p < \frac{2^{62}}{2^{128}}$.

So more generally, $q(x)$ can be defined as the polynomial resulting from multiplying a number of subgroup polynomials among the previously mentioned 163 subgroups such that the number of roots or the degree of the resulting polynomial lies in the interval $(2^{32}, 2^{62})$. The largest subgroup with order less than 2^{62} has order $t_1 t_2 t_4 t_5 t_7 t_8 \approx 2^{61.98}$ and its elements are the roots of $x^{t_1 t_2 t_4 t_5 t_7 t_8} + 1$. Choosing any product of subgroup polynomials $q_i(x)$ such that $q(x) = \prod_i q_i(x)$ have a number of roots that lies in the interval $(2^{32}, 2^{62})$ will give us the required forgery probability p needed for POET, $\frac{2^{32}}{2^{128}} < p < \frac{2^{62}}{2^{128}}$.

Weak Key Recovery By performing binary search on the roots of our forgery polynomial $q(x)$, we can easily recover the weak key by testing whether the hashing polynomial $q(x) =$

$\prod_{i=1}^{j}(x-\alpha_i)$, where $\{\alpha_1, \cdots, \alpha_j\}$ are the current binary searched roots of $q(x)$, yields a successful distinguisher on the polynomial hashing scheme under consideration. This will cost only n queries to the POET scheme if the forgery polynomial $q(x)$ has 2^n roots.

4 Impact of weak keys on POET

Having seen that the classes of weak keys described in Section 3.4 are present in POET, we discuss the implication of having one such key as the universal hash key L_{top}. Since POET allows nonce-reuse, we consider nonce-repeating adversaries, i. e. for our purposes, the nonce will be fixed to some constant value for all encryption and verification queries.

4.1 Observations

Observation 1 (Collisions in g_t imply tag collisions). *Let $M = M_1, \ldots, M_m$ and $M' = M'_1, \ldots, M'_m$ be two distinct messages of m blocks length such that $g_t(M) = g_t(M')$ or $g_t(M_1, \ldots, M_\ell) = g_t(M'_1, \ldots, M'_\ell)$ with $\ell < m$ and $M_i = M'_i$ for $i > \ell$. This implies a collision on POET's internal state X_i, Y_i for $i = m$ or $i = \ell$ respectively, and therefore equal tags for M and M'.*

We note that such a collision also allows the recovery L_{top} by means of the key search procedure outlined in Sect. 3.4.

Observation 2 (Knowing L_{top} implies knowing L_{bot}). *Once the first hash key L_{top} is known, the second hash key L_{bot} can be determined with only two 2-block queries: Choose arbitrary M_1, M_2, ∇_1 with $\nabla_1 \neq 0$ and obtain the encryptions of the two 2-block messages M_1, M_2 and M'_1, M'_2 with $M'_1 = M_1 \oplus \nabla_1, M'_2 = M_2 \oplus \nabla_1 \cdot L_{top}$. Denote $\Delta_i = C_i \oplus C'_i$. Then we have the relation $\Delta_1 \cdot L_{bot} = \Delta_2$, so $L_{bot} = \Delta_1^{-1} \cdot \Delta_2$.*

It is worth noting that this procedure works for arbitrary L_{bot}, and is in particular not limited to L_{bot} being another root of the polynomial q.

We now describe the impact of these observations in detail. Concerning the concept of weak keys, our attack scenario is based upon the universal approach of [6].

4.2 A generic forgery

In the setting of [6], consider an arbitrarily chosen polynomial $q(x) = \sum_{i=1}^{m-1} q_i x^i$ of degree $m-1$ and some message $M = M_1 \| \cdots \| M_{m-1} \| M_m$. Write $Q = q_1 \| \cdots \| q_{m-1}$ and define $M' \stackrel{\text{def}}{=} M + Q$ with Q zero-padded as necessary. For a constant nonce (1-block header) H, denote ciphertext and tag corresponding to M by $C = C_1, \ldots, C_m$ and T, and ciphertext and tag corresponding to $M' = M + Q$ by $C' = C'_1, \ldots, C'_m$ and T', respectively.

If some root of q is used as the key L_{top}, we have a collision between M and $M' = M + Q$ in the polynomial hash evaluation after $m-1$ blocks:

$$\tau {L_{\text{top}}}^{m} + \sum_{i=1}^{m-1} M_i {L_{\text{top}}}^{m-i} = \tau' {L_{\text{top}}}^{m} + \sum_{i=1}^{m-1} M'_i {L_{\text{top}}}^{m-i}$$

This implies $X_{m-1} = X'_{m-1}$ and therefore $Y_{m-1} = Y'_{m-1}$. Since the messages are of equal length, $S = S'$ and we also have a collision in X_m and Y_m. It follows that $C_m = C'_m$. Furthermore, since $\tau = \tau'$, the tag T is colliding as well. Since then M and $M + Q$ have the same tag, $M + Q$ is a valid forgery whenever some root of q is used as L_{top}.

Note that both M and the forged message will be m blocks long.

4.3 Universal weak-key forgeries for POET

In this section, we describe that weak keys enable universal forgeries for POET under the condition that the order of the weak key is smaller than the maximal message length in blocks. Note that this is the case for all polynomials described in Section 3.4.

For obtaining universal forgeries, we first use the polynomial hash collision described above to recover the weak keys L_{top} and L_{bot}, and then recover τ, which is equal to the initial states X_0 and Y_0, under the weak key assumption.

Recovering $\boldsymbol{\tau}$ Suppose that we have recovered the weak keys $L_{\text{top}}$and L_{bot}. Now our goal is to recover the secret $X_0 = Y_0 = \tau$. We know that

$$X_i = \tau L_{\text{top}}^i + M_1 L_{\text{top}}^{i-1} + M_2 L_{\text{top}}^{i-2} + \cdots + M_i$$

and

$$X_{i+j} = \tau L_{\text{top}}^{i+j} + M_1 L_{\text{top}}^{i+j-1} + M_2 L_{\text{top}}^{i+j-2} + \cdots + M_{i+j}.$$

Now if L_{top} has order j , i.e. $L_{\text{top}}^j = \text{Identity}$, then we get $X_i = X_{i+j}$ by constructing $M_{i+1}, \cdots, M_{i+j}$ such that

$$M_{i+1} L_{\text{top}}^{j-1} + M_{i+2} L_{\text{top}}^{j-2} + ... + M_{i+j} = 0.$$

The easiest choice is to set $M_{i+1} = M_{i+2} = \cdots = M_{i+j} = 0$. This gives us $Y_i = Y_{i+j}$. Now equating the following two equations and assuming that $L_{\text{bot}}^j \neq \text{Identity}$,

$$Y_i = \tau L_{\text{bot}}^i + C_1 L_{\text{bot}}^{i-1} + C_2 L_{\text{bot}}^{i-2} + \cdots + C_i$$

and

$$Y_{i+j} = \tau L_{\text{bot}}^{i+j} + C_1 L_{\text{bot}}^{i+j-1} + C_2 L_{\text{bot}}^{i+j-2} + \cdots + C_{i+j}$$

we get

$$\tau = (C_1 L_{\text{bot}}^{i-1} + C_2 L_{\text{bot}}^{i-2} + \cdots + C_i + C_1 L_{\text{bot}}^{i+j-1} + C_2 L_{\text{bot}}^{i+j-2} + \cdots + C_{i+j})(L_{\text{bot}}^i + L_{\text{bot}}^{i+j})^{-1}.$$

Querying POET's block cipher $\boldsymbol{E_K}$ One can see from Fig. 3 that once we know L_{top}, L_{bot} and τ, we can directly query POET's internal block cipher without knowing its secret key K. internal block cipher, i.e. we want to compute $E_K(x)$. Now from Fig. 3, we see that the following equation holds

$$E_K(\tau L_{\text{top}} \oplus M_1) = C_1 \oplus \tau L_{\text{bot}},$$

therefore

$$E_K(x) = C_1 \oplus \tau L_{\text{bot}}.$$

If M_1 was the last message block, however, we would need the encryption $S = E_K(|M|)$. Therefore we have to extend the auxiliary message for the block cipher queries by one block, yielding the following:

Observation 3 (Querying POET's block cipher). *Knowing L_{top}, L_{bot} and τ enables us to query POET's internal block cipher without the knowledge of its secret key K. To compute $E_K(x)$ for arbitrary x, we form a two-block auxiliary message $M'_1 = (x \oplus \tau L_{top}, M'_2)$ for arbitrary M'_2 and obtain its POET encryption as C'_1, C'_2. Computing $E_K(x) := C'_1 \oplus \tau L_{bot}$ then yields the required block cipher output.*

This means that we can produce valid ciphertext blocks $C_1, \ldots, C_{\ell_M}$ and (if necessary) partial tags T^α for any desired messages, by simply following the POET encryption algorithm using the knowledge of $L_{\text{top}}, L_{\text{bot}}, \tau$ and querying POET with the appropriate auxiliary messages whenever we need to execute an encryption E_K. Note that this also includes the computation of $S = E_K(|M|)$. A complete example is given in Sect. A in the appendix.

Generating the final tag In order to generate the second part of the tag T^β (see Fig. 2), which is the full tag T for integral messages, we use the following procedure.

We know the value of X_{ℓ_M} for our target message M from the computation of C_{ℓ_M}. If we query the tag for an auxiliary message M' with the same $X'_{\ell_{M'}}$, the tag for M' will be the valid tag for M as well, since having $X'_{\ell_{M'}} = X_{\ell_M}$ means that $Y'_{\ell_{M'}} = Y_{\ell_M}$ and consequently $T^{\beta\prime} = T^\beta$.

Therefore, we construct an auxiliary one-block message $M' = (X_{\ell_M} \oplus E_K(|M'|) \oplus \tau L_{\text{top}}$ and obtain its tag as T' (computing the encryption of the one-block message length by querying E_K as above). By construction $X'_1 = X_{\ell_M}$, so T' is the correct tag for our target message M as well.

By this, we have computed valid ciphertext blocks and tag for an arbitrary message M by only querying some one- or two-block auxiliary messages. This constitutes a universal forgery.

We finish by noting that in case a one- or two-block universal forgery is requested, we artificially extend our auxiliary messages in either the final tag generation (for one-block targets) or the block cipher queries (for two-block messages) with one arbitrary block to avoid having queried the target message as one of our auxiliary message queries.

4.4 Further forgery strategies

Since the universal forgery of the previous section relies on having a weak key L_{top} with an order smaller than the maximum message length for recovering τ, we describe two further forgery strategies that are valid for any weak key, regardless of its order.

Constructing shorter (blind) forgeries Having generated a polynomial hash collision, and therefore recovered the universal hash keys L_{top} and L_{bot}, we can freely produce blind forgeries for any ciphertext-tag pair of at least 2 blocks length. Suppose we have a ciphertext $C = C_1, \ldots, C_m$ with corresponding tag T for $m \geq 2$. Then T is also a valid tag for $C' = (C_1 \oplus \Delta, C_2 \oplus \Delta \cdot L_{\text{bot}}, C_3, \ldots, C_m)$ and the same nonce, since during the decryption process, we have $Y'_2 = C_2 \oplus \Delta \cdot L_{\text{bot}} \oplus (C_1 \oplus \Delta \oplus \tau \cdot L_{\text{bot}}) \cdot L_{\text{bot}} = C_2 \oplus (C_1 \oplus \tau \cdot L_{\text{bot}}) \cdot L_{\text{bot}} = Y_2$. Therefore $X'_2 = X_2$ as well, and this collision is preserved by having $C'_i = C_i$ for $i > 2$.

Constructing meaningful (targeted) forgeries We can also leverage collisions in the polynomial hash to produce targeted forgeries with complete control over the differences in the first $m - 2$ message blocks with a complexity of only two encryption queries per forgery.

The length of these queries is one block longer or shorter than the length of the message we want to provide a forgery for, and can be as short as two blocks. Being able to produce forgeries for arbitrary messages with *chosen* differences in the first $m-2$ message blocks already comes close to a universal forgery.

We first describe the procedure for the case of m-block messages with $m \geq 3$ and deal with $m = 2$ later.

Let $m \geq 3$, $M = M_1, \ldots, M_{m-1}, M_m$ denote the target message, $(C_1, \ldots, C_m; T)$ its encryption and tag and $\nabla_1 \neq 0, \ldots, \nabla_{m-2} \neq 0$ the desired differences in $M_1, \ldots, M_{m-2}$. We then produce a valid ciphertext with equal tag T for $M_1 \oplus \nabla_1, M_2 \oplus \nabla_2, \ldots, M_{m-1} \oplus \nabla_{m-1}, M_m$, with uncontrollable ∇_{m-1}.

Step 1: Recovering L_{top}. We first note that the collisions in C_m and T from the generic forgery can be used to detect the collision in $g_t(X)$ and therefore whether a root of q was used as L_{top}. We can then use the key search algorithm outlined in Sect. 3.4 to recover the value of L_{top} with about $128 - \log_2(m) + 1$ verification queries.

Step 2: Querying for prefix. Once L_{top} is known, we can use this to query for a prefix of our forged message as follows. Define

$$\nabla_{m-1} \overset{\text{def}}{=} \begin{cases} \nabla_1 \cdot L_{\text{top}} & \text{if } m = 3 \\ \nabla_1 \cdot (L_{\text{top}})^{m-2} \oplus \cdots \oplus \nabla_{m-2} \cdot L_{\text{top}} & \text{if } m > 3. \end{cases}$$

Form $m-1$-block messages $M_1, \ldots, M_{m-1}$ and $M'_1, \ldots, M'_{m-1}$ with $M'_i \overset{\text{def}}{=} M_i \oplus \nabla_i$, and obtain their encryptions $C_1, \ldots, C_{m-1}$ and $C'_1, \ldots, C'_{m-1}$. Denote the ciphertext differences by $\varDelta_i \overset{\text{def}}{=} C_i \oplus C'_i$. Note that ∇_{m-1} is chosen to eliminate the differences introduced by the previous message blocks, yielding $X_{m-1} = X'_{m-1}$ and therefore also $Y_{m-1} = Y'_{m-1}$, a collision on the internal state of POET. This situation is illustrated in Fig. 3.

Step 3: Constructing the forgery. The knowledge of the "right pair" $(M_1, \ldots, M_{m-1})$ and $(M'_1, \ldots, M'_{m-1})$ for our internal state collision differential now enables us to construct the desired forgery. Query POET on the target message $M = (M_1, \ldots, M_{m-1}, M_m)$ and obtain ciphertext $C = (C_1, \ldots, C_m)$ and tag T. Then $(C_1 \oplus \varDelta_1, \ldots, C_{m-1} \oplus \varDelta_{m-1}, C_m; T)$ is a valid ciphertext-tag pair for $(M_1 \oplus \nabla_1, \ldots, M_{m-1} \oplus \nabla_{m-1}, M_m)$. Since this message was not queried before, this constitutes a valid forgery.

Constructing two-block forgeries. If the target message is two blocks long, we cannot use the above procedure since we need at least a two-block prefix query to achieve the internal state collision. For $m = 2$, we would then already have queried the message forged in Step 3 in Step 2. We can however follow an entirely analogous procedure by simply extending the queries in Step 2 by one arbitrary block Z. Let ∇_1 be the chosen difference for the first message block. Compute L_{top} as described in Step 1. In Step 2, we then obtain the encryption of (M_1, M_2, Z) as (C_1, C_2, C_Z) and $(M_1 \oplus \nabla_1, M_2 \oplus \nabla_1 \cdot L_{\text{top}}, Z)$ as (C'_1, C'_2, C'_Z), and then construct the forgery in Step 3 as (C'_1, C'_2).

5 Weak keys and OPERM-CCA security

POET is designed as a decryption-misuse-resistant online cipher [5], meaning that modifying the i-th ciphertext block C_i should result in random changes to all plaintext blocks

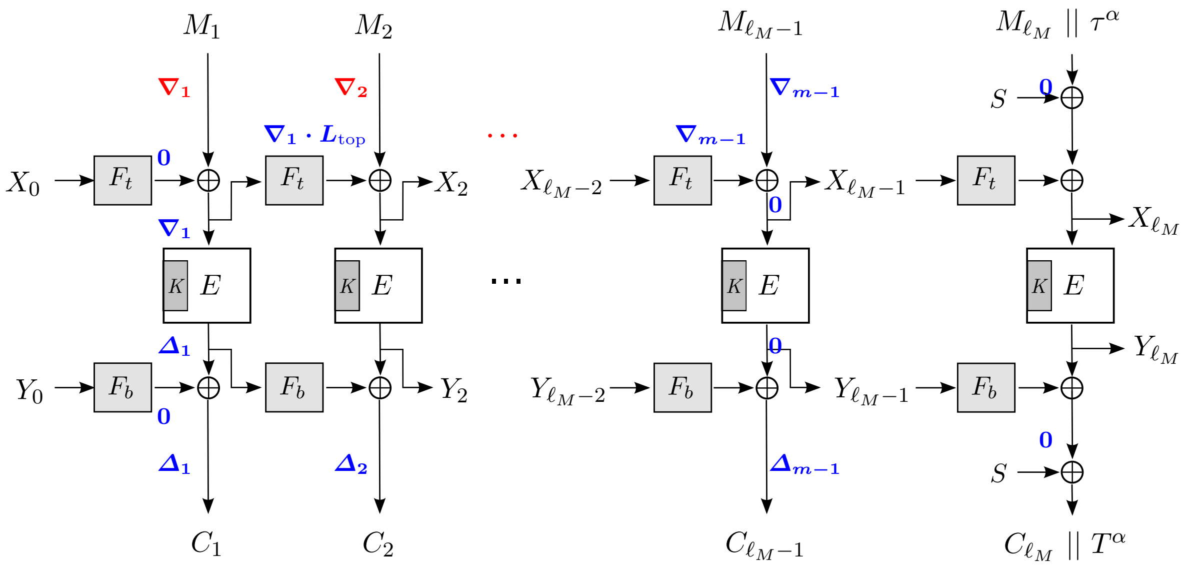

Fig. 3: Constructing targeted forgeries for POET. Freely chosen differences are indicated in red, uncontrolled differences in blue.

$M_i, M_{i+1}, \ldots$. This property is called OPERM-CCA security [5]. In the following we show that if a weak key is used as L_{top}, POET's underlying online cipher POE does not provide OPERM-CCA security (noting that, according to the results of [6], every key can be weak). As outlined in the previous sections, use of a weak L_{top} allows us to recover both L_{top} and L_{bot}. Furthermore, we assume that a fixed nonce is being used.

A distinguisher. Let $M_1 \neq M_2$ two different message blocks. We obtain the encryption of M_1, M_2 as C_1, C_2. We then choose an arbitrary difference $\Delta \neq 0$ and ask for the decryption of the two-block ciphertext

$$C_1 \oplus \Delta, C_2 \oplus \Delta \cdot L_{\text{bot}}$$

which we denote M_1', M_2'. We then verify if

$$M_2 \oplus M_2' \overset{?}{=} (M_1 \oplus M_1') \cdot L_{\text{top}} \qquad (1)$$

If this equation is fulfilled, our distinguisher concludes that the POE online cipher has been used, otherwise a random online permutation. The probability of a false positive in (1) is around 2^{-n}.

Constructing ciphertexts for specific plaintext blocks. Suppose we obtain the encryption of a message $M = M_1, M_2, \ldots, M_m$ as $C = C_1, C_2, \ldots, C_m$ with $m \geq 3$. Analogous to the distinguisher above, by constructing

$$C' = C_1 \oplus \Delta, C_2 \oplus \Delta \cdot L_{\text{bot}}, C_3, \ldots, C_m$$

for any $\Delta \neq 0$, we have constructed a new ciphertext with known message blocks starting from $m = 3$, which would not be possible for a CCA-secure online permutation. We also note that this strictly speaking only requires the knowledge of L_{bot} (which in the weak key scenario however requires a weak L_{top}).

References

1. Farzaneh Abed, Scott Fluhrer, John Foley, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. The POET Family of On-Line Authenticated Encryption Schemes. Submission to the CAESAR competition, 03 2014.
2. Neils Ferguson. Authentication weaknesses in GCM. Comments submitted to NIST Modes of Operation Process, 2005.
3. Helena Handschuh and Bart Preneel. Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer, 2008.
4. Antoine Joux. Authentication Failures in NIST version of GCM. Comments submitted to NIST Modes of Operation Process, 2006.
5. David McGrew, Scott Fluhrer, Stefan Lucks, Christian Forler, Jakob Wenzel, Farzaneh Abed, and Eik List. Pipelineable On-Line Encryption. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption, FSE 2014*, Lecture Notes in Computer Science, page 24. Springer-Verlag, 2014. to appear.
6. Gordon Procter and Carlos Cid. On Weak Keys and Forgery Attacks against Polynomial-based MAC Schemes. In Shiho Moriai, editor, *Fast Software Encryption, FSE 2013*, Lecture Notes in Computer Science, page 14. Springer-Verlag, 2013. to appear.
7. Markku-Juhani Olavi Saarinen. Cycling Attacks on GCM, GHASH and Other Polynomial MACs and Hashes. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 216–225. Springer, 2012.

A Universal forgeries: an example

Suppose that we want to generate the first ciphertext block C_1 of the message $M = M_1||\cdots||M_{l_M}$. Then we query POET for a message $M' = M'_1||M'_2$ where M'_i are chosen as follows.

To find the ciphertext block C_1, we need to query POET's block cipher for the encryption of $M_1 \oplus \tau L_{\text{top}}$. To do this we set $X'_2 = M_1 \oplus \tau L_{\text{top}}$. Now since $X'_2 = \tau L^2_{\text{top}} \oplus M'_1 L_{\text{top}} \oplus M'_2$, then

$$M_1 \oplus \tau L_{\text{top}} = \tau L^2_{\text{top}} \oplus M'_1 L_{\text{top}} \oplus M'_2$$

Setting $M'_1 = 0$ and $M'_2 = M_1 \oplus \tau L_{\text{top}} \oplus \tau L^2_{\text{top}}$ gives us the ciphertext blocks $C' = C'_1||C'_2$. Now $C_1 = \tau L_{\text{bot}} \oplus Y'_2$. But $Y'_2 = \tau L^2_{\text{bot}} \oplus C'_1 L_{\text{bot}} \oplus C'_2$. Therefore

$$C_1 = \tau L_{\text{bot}} \oplus \tau L^2_{\text{bot}} \oplus C'_1 L_{\text{bot}} \oplus C'_2$$

To generate the other ciphertext blocks C_i's, where $i \geq 2$, we ask for the encryptions $y_i = E_K(X_i)$ where $X_i = \tau L^i_{\text{top}} \oplus M_1 L^{i-1}_{\text{top}} \oplus M_2 L^{i-2}_{\text{top}} + \cdots + M_i$ for each i by performing i queries to the POET scheme on the message $M' = M'_1||M'_2$ where $M'_1 = X_i \oplus \tau L_{\text{top}}$. Then $C_i = Y_{i-1} L_{\text{bot}} \oplus Y_i$.

For the generation of the final tag T^β, we use the procedure of Sect. 4.3.