Holder-of-key threshold access token for anonymous data resources (Proof Of Security)

Micael Pedrosa
University of A Coruña
Coruña, Spain
University of Aveiro
Aveiro, Portugal
micaelpedrosa@ua.pt

Rui Lebre
University of A Coruña
Coruña, Spain
University of Aveiro
Aveiro, Portugal
ruilebre@ua.pt

Carlos Costa
University of Aveiro
Aveiro, Portugal
carloscosta@ua.pt

I. Proof of Security

This section demonstrates the security of the proposed method. It is assumed that honest nodes verify if elliptic curve points are in the correct group and if scalars are in the correct range, avoiding an entire class of issues [1].

A. Token forgery

This section proves that the token is non-malleable and cannot be forged within the defined threat model.

Lemma 1. *Let $i \in [1, t+1] \cap \mathbb{Z}$ and (y, m, a, a_i) be unknown secrets, where (y_i, m_i, a_i) are the required shares to recover (y, m, a). Then, Y_a and M_a can only be constructed via $\mathcal{L}^i(y_i) \times A \mapsto Y_a$ and $\mathcal{L}^i(m_i) \times A \mapsto M_a$.*

Proof: Trivially, since (y, m, a) and a_i are unknown secrets, then, both $y \cdot a \times G \mapsto Y_a$ and $m \cdot a \times G \mapsto M_a$ calculations cannot be performed. We can only get (Y_a, M_a) using the shares (y_i, m_i) and the parameterized point A.

Lemma 2. *t compromised shares from the set $\{m_i : i \in [1, t+1] \cap \mathbb{Z}\}$ where $\mathcal{L}^i(m_i) \mapsto m$, cannot force m to a known value.*

Proof: An adversary with t compromised nodes can try a rogue key attack [2] for a rushing adversary. These nodes are allowed to select their $m_i^{'}$ shares after observing the M_1 value of the honest node (via a rogue client), trying to force a known m. The following is the result of the rogue Lagrange interpolation:

$$l_1(0) \times M_1 + \sum_{i=2}^{t+1} l_i(0) \cdot m_i^{'} \times G \mapsto m \times G \tag{1}$$

Assuming that the adversary is unable to guess m_1 from $m_1 \times G \mapsto M_1$, the exact combination of $m_i^{'}$ values that produce m cannot be known due to the intractable DLP for M_1. If we define $x = \sum_{i=2}^{t+1} l_i(0) \cdot m_i^{'}$ and $M_1^{'} = l_1(0) \times M_1$ then, the DLP is reduced to $(m - x) \times G \mapsto M_1^{'}$.

Lemma 3. *Assuming a maximum of t compromised shares and a rogue client, m is distinct for each session s with a high probability.*

Proof: An adversary can force m to be reused if it can compromise $t + 1$ shares, since it can directly chose all m_i shares for $\mathcal{L}^i(m_i) \mapsto m$. However, for a maximum of t compromised shares, there is always one m_i that depends on an hash result from (seq, tms, P_t). Since the strictly increasing seq is controlled by the honest node, and due to the second-preimage resistance, it is hard for a rogue client to force the inputs (tms, P_t) such that the hash function results in the required m_i. If m has enough bits, the probability of deriving the same m using the distinct and honest m_i value should be negligible. Note that, the tms is mainly present to minimise overflows of the seq value in real implementations. Moreover, the range of the tms value should be checked against the internal clock, and P_t should be a valid client key.

Theorem 1. *Both Y_a and M_a are secret points, where M_a is distinct (with high probability) for each token T_k.*

Proof: From lemma 1 both results are intractable Diffie-Hellman computations that can only be resolved via Lagrange interpolations. However, both interpolations that can compute those points are never executed independently. Even if we assume that (r, k) are compromised and remove those values from the token, T_k is preserved as a linear combination of the form $(Y_a + x \times M_a)$, without exposing Y_a or M_a individually. Both secrets are always present in the token since $a \neq 0$, $y \neq 0$ and $m \neq 0$ with high probability, and also because $(x = k \cdot c) \neq 0$.

M_a cannot be recovered from subtracting two different tokens. Assuming distinct values for m (with high probability) from lemma 3, then M_a values are also distinct for each token. This leads to the impossibility of a rogue client to get M_a from two different requested tokens, by performing $(T_{1k} - T_{2k}) = (x_1 - x_2) \times M_a$ and recovering M_a, assuming that x_1 and x_2 are known. This recovering mechanisms cannot be used since M_a is always distinct, leading to $(T_{1k} - T_{2k}) = x_1 \times M_{1a} - x_2 \times M_{2a}$ or $(T_{1k} - T_{2k}) = x \times (M_{1a} - M_{2a})$ if $x_1 = x_2$. Both M_{1a} or M_{2a} cannot be recovered independently. Consequently, Y_a cannot be recovered by this process if M_a is unknown.

From lemma 2, m cannot be tampered with. Yet, if a rogue client is able to chose any values for $(A_r^{'}, A_{k \cdot c}^{''})$ such that the respective secrets $(a^{'}, a^{''})$ are known, the previously defined linear combination is reduced to $a^{'} \times Y + a^{''} \cdot x \times M$. Since

Y and M are known, by tampering with one value $(a^{'}, a^{''})$ at a time the adversary can recover Y_a and M_a. For instance, by subtracting $a^{''} \cdot x \times M$ from the equation, one can recover Y_a if $a^{'}$ is left unchanged, where $a^{'} = a$. To prevent this attack, the following verifications are required at each node: $e(A_r, G^{\dagger}) \stackrel{?}{=} e(R, A^{\dagger})$ and $e(A_{k \cdot c}, G^{\dagger}) \stackrel{?}{=} e(K_c, A^{\dagger})$. These assures that $A_r^{'}$ and $A_{k;c}^{''}$ are always generated from the base point A and that $(a^{'}, a^{''})$ are unknown.

Lemma 4. *A valid token T_k cannot be produced by randomly selecting M, M_k and π, or by using these parameters from other valid tokens.*

Proof: From the *VerifToken* function, the verification at *4)* is equivalent to $a^{-1} \times T_k \stackrel{?}{=} \pi + c \times M_k$. However, since a^{-1} is unknown, this verification can only be performed using bilinear pairings. Moreover, if we check that $H_p(M||M_k||\pi) \stackrel{?}{=} c$, this is the same verification procedure that is used in the Schnorr's signatures. The main difference is that a^{-1} (that should correspond to the r_σ value in σ, being that $a^{-1} \equiv r_\sigma$) is a fixed value, and $T_k \equiv G$ is a dynamic base point. In this way, the signature verification can be done with Pairing-Based Cryptography (PBC) without requiring to know a. Nonetheless, the presence of a is verified with $A^{\dagger}$. The σ_k signature verification binds M as the base point for M_k. Any linear changes in (M, M_k, π) produces non-linear changes in c with the same security proofs as the ones used in Schnorr's signatures. Note that, values such as $(M_{1a} - M_{2a})$, are not valid since c has a different committed value. Furthermore, since Y_a and M_a are unknown from theorem 1, one cannot produce T_k from a combination of those secrets. For instance, a valid $T_k = r \times Y_a + c \cdot k \times M_a$ would be possible if r and k were known.

Lemma 5. *A valid token T_k cannot be produced from another invalid token.*

Proof: Note that c is used blindly and T_k is not included in it as a commitment value. In this context an invalid $T_k^{'} = \pi_a^{'} + c^{'} \times M_{k \cdot a}^{'}$ can be produced by submitting a fake $c^{'}$ such that $c = c^{'} \cdot c^{''}$, where $H_p(M||M_k||\pi) \mapsto c$ contains the π target that we want to attack. The invalid $T_k^{'}$ could be transformed to a valid one via $c^{''} \times T_k^{'}$, if $c^{''} \times \pi^{'} \mapsto \pi$. However this would require to have a profile with a $R^{'}$ value such that $c^{''} \times R^{'} \mapsto R$. Since $r^{'}$ is not controlled by the rouge client or a single node, this reduces to the DLP for $R^{'}$.

The second alternative is to attack $M_{k \cdot a}^{'}$, such that when it is used in the token computation results in $c^{''} \cdot m \times A_{k \cdot c}^{'} = \pi_a - c^{''} \times \pi_a + c \times M_{k \cdot a}$. From the token transformation, $c^{''} \times \pi_a$ is eliminated, resulting in a valid π_a. However, to forge a correct value for $A_{k \cdot c}^{'} = m^{-1} \cdot (c^{''-1} - 1) \times \pi_a + c^{''-1} \cdot c \times A_k$, one needs to solve the DLP for the unknown m value. Moreover, $A_{k \cdot c}^{'}$ requires a known scalar $k \cdot c$ that depends on m in order to pass the $e(A_{k \cdot c}, G^{\dagger}) \stackrel{?}{=} e(K_c, A^{\dagger})$ validation.

Theorem 2. *A valid T_k cannot be forged within the defined threat model.*

Proof: From theorem 1, since (Y_a, M_a) are secrets, only $t+1$ nodes can produce a $(Y_a + x \times M_a)$ combination with an interlaced x. From lemmas 4 and 5 we conclude that T_k is non-malleable. A valid or invalid $T_k^{'}$ cannot be transformed into a valid T_k.

B. Pseudonymity disclosure

This section proves that the $I \mapsto \pi$ mapping is not revealed from the token or any other public parameters. Note that k can disclose the mapping, but this is not public parameter. We assume that the owner of an authorized k key already knows the mapping.

Lemma 6. *The embedded information in session s (seq, tms, P_t) does not correlate with the token public values $(T_k, M_k, M, \pi, c, \sigma_k)$. Consequently, the session does not reveal the $I \mapsto \pi$ mapping.*

Proof: Even in the presence of distinct values such as (seq, tms, P_t), the resulting interpolations $\mathcal{L}^i(m_i) \times G \mapsto M$ and $\mathcal{L}^i(y_i) \times R \mapsto \pi$ does not contain any information of those values. M_k is derived from M and T_k is derived from (M_k, π), with no correlation with s. By definition c must not contain any information from the session. Since the session is not used in the *Access Request* and the token has no direct correlation with the session, the session does not reveal the $I \mapsto \pi$ mapping.

Lemma 7. *Values associated with the token (T_k, M_k, M, π, c) do not correlate with the public parameters $(R, A_r, K_c, A_{k \cdot c})$ that are associated with the profile I.*

Proof: The results from $c^{-1} \times K_c \mapsto K$ and $c^{-1} \times A_{k \cdot c} \mapsto A_k$ are also associated with the token, but maintained in the private space of the client. Without prior knowledge of (K, A_k) that are in the private space of the client, the payload c cannot provide a correlation with $(K_c, A_{k \cdot c})$, in the public space of federated nodes. The main vulnerability is to get correlations via bilinear pairings. We will use subscripts to include scalars in points such as $m \times A^{\dagger} \mapsto A_m^{\dagger}$. Important pairing correlations are in the list:

1) $e(T_k, G^{\dagger}) = e(\pi, A^{\dagger}) \cdot e(M_k, A^{\dagger})^c$
2) $e(A_r, G^{\dagger}) = e(R, A^{\dagger}) = e(A, R^{\dagger})$
3) $e(A_k, G^{\dagger}) = e(K, A^{\dagger}) = e(A, K^{\dagger})$
4) $e(M_k, G^{\dagger}) = e(K, M^{\dagger}) = e(M, K^{\dagger})$
5) $e(M_k, A^{\dagger}) = e(K, A_m^{\dagger}) = e(M, A_r^{\dagger})$
6) $e(\pi, G^{\dagger}) = e(R, Y^{\dagger}) = e(Y, R^{\dagger})$
7) $e(\pi, A^{\dagger}) = e(A_r, Y^{\dagger}) = e(A, Y_r^{\dagger})$
8) $e(\pi, A^{\dagger}) = e(Y, A_r^{\dagger}) = e(R, A_y^{\dagger})$

We may recover useful information from *4)*, *5)*, *6)*, *7)* and *8)*; however, we only have access to $(G^{\dagger}, A^{\dagger})$ points in the $\mathbb{G}_2$ group. Also, there is no homomorphism $\phi : \mathbb{G}_2 \mapsto \mathbb{G}_1$ that can produce the required points. From this result we should notice that $Y^{\dagger}$ is also an important secret to retain anonymity.

The interpolation that reveals this secret $\mathcal{L}^i(y_i \times G^\dagger) \mapsto Y^\dagger$ is never executed.

Theorem 3. *The $I \mapsto \pi$ mapping can only be disclosed by $t+1$ multiparty computations or by the owner of k, within the defined threat model.*

Proof: From lemmas 6 and 7 we conclude that there are no direct correlations disclosing the mapping. From the DLP hardness assumption and information-theoretic security of Shamir's Secret Sharing, t shares are not enough to produce the result from $\mathcal{L}^i(y_i \times R) \mapsto \pi$, where $R \mapsto I \mapsto \pi$. With r one could directly map $r \times Y \mapsto \pi$, but r is not known. With k one can connect $c \times K \mapsto K_c$, where $K \mapsto K_c \mapsto R \mapsto I \mapsto \pi$.

REFERENCES

[1] A. Antipa, D. Brown, A. Menezes, R. Struik, and S. Vanstone, "Validation of elliptic curve public keys," in *International Workshop on Public Key Cryptography*. Springer, 2003, pp. 211–223.

[2] T. Ristenpart and S. Yilek, "The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2007, pp. 228–245.