Proving Security of Tor's Hidden Service Identity Blinding Protocol
Tor Project Tech Report 2013-12-001

Nicholas Hopper

1 Introduction

Tor's Hidden Services allow a server to offer a service anonymously through the Tor network. Hidden Services are associated with an *identity key* and need to publish *descriptors* that allow clients to locate *introduction points* to which the hidden service has already established anonymous circuits. As Biryukov *et al.* show [2], in the current Hidden Service design, a Tor Node that is in position to publish the descriptor for a hidden service can learn the identity key of the service and launch further attacks that could allow it to de-anonymize the service. This report analyzes the security of a new design proposed by Robert Ransom [3] and specified in full as Tor Proposal 224 to eliminate these attacks.

2 Preliminaries

2.1 Ed25519

The design makes use of the Ed25519 signature scheme [1]. Ed25519 is defined over an Edwards-form elliptic curve taken modulo the prime $2^{255} - 19$. We follow the notation of [1] in our description: Points on the curve are represented by capital letters, scalars by lower case letters; for any point P on the curve, we denote its compressed encoding as $\underline{P}$; B is the base point on the curve specified by [1], $\ell \approx 2^{252}$ is the order of B (the least integer such that $\ell B = 0$) and $b = \lfloor \log_2 \ell \rfloor + 3 = 255$. The scheme also makes use two cryptographic hash functions, modeled as random oracles: $H_r : \{0,1\}^* \to \{0,1\}^{2b}$ is the "randomizing" hash and $H_m : \{0,1\}^* \to \{0,1\}^{2b}$ is the "compressing" hash. Key generation, signing and verification are similar (but not identical) to Schnorr signatures:

Key generation. Choose a random element $a \in_R \mathbb{Z}_\ell$, and a random string $k \in_R \{0,1\}^b$. The (public) verification key is $A = aB$ and the (secret) signing key is the pair (a, k).

Signing. To sign the message m, we compute $r = H_r(k, m)$, set $R = rB$, compute $c = H_m(\underline{R}, \underline{A}, m)$, and set $s = r + ca \bmod \ell$. The signature is the pair $(\underline{R}, s)$.

Verification To verify the signature $(\underline{R}, s)$ on message M with verification key A, we compute $h = H_m(\underline{R}, \underline{A}, M)$ and check that $hA + R = sB$.

2.2 Hardness assumptions

Ed25519 Unforgeability We assume that Ed25519 signatures are $(t, q, \epsilon_{\mathsf{sig}})$-secure against existential forgery under chosen message attack: for every attack with complexity at most t, that can query adaptively for signatures on up to q messages $m_1, \ldots, m_q$, the probability of producing a new message $m \notin \{m_1, \ldots, m_q\}$ and a correct signature $(\underline{R}, s)$ on m is bounded by ϵ_{sig}. For an arbitrary algorithm $\mathcal{A}$, we let $\mathsf{SForge}_{\mathcal{A}}$ denote the outcome (0 for failure, 1 for success) of this chosen message attack against Ed25519.

AES-CTR indistinguishability We assume that AES-CTR mode encryption provides $(t, \epsilon_{\mathsf{pre}})$-pseudorandomness: for every attack A with complexity at most t that produces a challenge plaintext m, for uniformly chosen $K \in \{0,1\}^{128}$, we have

$$\left|\Pr[A(E_K(m)) = 1] - \Pr[A(U_{|m|+128}) = 1]\right| \leq \epsilon_{\mathsf{pre}} \ ,$$

Where U_λ denotes a random variable that takes on a uniform random value in $\{0,1\}^\lambda$.

2.3 Random Oracles

The scheme will make use of several cryptographic hash functions H_r, H_m, H_e and H_d, which we model as random oracles (i.e. "generic functions"). In practice they can be implemented using a single cryptographic hash function H, for example SHA512 or Keccak, and separating them with distinct prefixes, for example $H_r(x) = H(\text{“}\texttt{randomize}\text{”}||x)$, $H_m(x) = H(\text{“}\texttt{compress}\text{”}||x)$, $H_e(x) = H(\text{“}\texttt{epoch}\text{”}||x)$ and $H_d(x) = H(\text{“}\texttt{descriptor}-\texttt{key}\text{”}||x)$.

3 Ransom's scheme

Ransom's scheme [3] seeks to prevent a Tor Hidden Service Directory node that does not know the identity key of a hidden service from learning the identity key from a descriptor or linking descriptors for the same identity key, while preventing forgery of descriptors. The scheme works by breaking time into epochs across which descriptors should be unlinkable. At each epoch, a hidden service generates an "epoch key" that can be derived from the identity key but cannot be used to derive the identity key, and this epoch key is used to encrypt and authenticate the signature. The entire scheme works as follows:

Identity Keys. Identity key pairs are chosen exactly as in Ed25519.

Epoch Keys. Let A be a public identity key (and (a, k) the corresponding signing key); then the epoch identity key for epoch t, A_t, is calculated by setting $s_t = H_e(t||A)$ and $A_t = s_t A$. The epoch signing key is $(s_t a, k)$.

Descriptors. Let info_t denote the plaintext information contained in the descriptor for epoch t (introduction point keys and other details needed to establish a connection to the

hidden service). The hidden service computes $k_t = H_d(t||\underline{A})$, $c_t = E_{k_t}(\mathsf{info}_t)$, and σ_t as the signature, under the epoch signing key for epoch t, on the message c_t, with one modification - when signing value m, the signer computes the randomness r as $r = H_r(k, t, m)$ to prevent reuse of r values across epochs. The full descriptor is A_t, c_t, σ_t.

Descriptor Verification. Given identity key A, epoch t and a descriptor A_t, c_t, σ_t, a client of the hidden service can verify the descriptor by parsing $\sigma_t = (\underline{R}, s)$, computing $s_t = H_d(t||\underline{A})$, $h = H_m(\underline{R}, \underline{A_t}, c_t)$, and checking that $A_t = s_t A$ and $s_t hA + R = sB$.

A Tor node that does not have the identity key A can still verify the descriptor against the epoch key by parsing $\sigma_t = (\underline{R}, s)$, computing $h = H_m(\underline{R}, \underline{A_t}, c_t)$ and checking that $hA_t + R = sB$.

Number of epochs. To prevent "ambiguity attacks" and simplify the security proof, we select a maximum number of epochs T and encode each epoch t using $\lceil \log_2 T \rceil$ bits. In practice, $T = 2^{20}$ would allow over 100 years of 1-hour epochs, far outlasting the likely security of Ed25519 in the face of Moore's law.

4 Security Definitions

The primary properties we would like from our new scheme are *unlinkability*: given a list of descriptors for the same service from several different epochs it should be infeasible to tell whether a valid descriptor from a distinct epoch is linked to the same identity key or another, without knowing the underlying identity key; and *descriptor unforgeability*: given an identity key and many valid descriptors for the identity it should be infeasible to produce a new, valid descriptor for the identity key and some epoch. Some remarks:

- Note that unlinkability also implies the weaker condition of "one-wayness:" an adversary that can recover the common identity key from a set of descriptors can easily recognize another valid descriptor for the same identity key.
- Furthermore, as will become clear below, unlinkability will also imply that without the identity key, an adversary learns no information about the contents of the descriptor.
- If we were "real cryptographers" we would define security in terms of a single game in which the adversary could choose among several identities to attack, and adaptively choose which goal to compromise. We stipulate that these definitions are slightly weaker, and forge ahead because, progress.

4.1 Unlinkability

We define the (t, q_E, q_H)-linkability experiment with adversary $\mathcal{A}$, $\mathsf{Link}_{\mathcal{A}}(t, q_E, q_H)$ as follows:

1. An identity key $A = aB$ and secret k are generated according to the description in section 3.

2. $\mathcal{A}$ makes q_E adaptive queries $(\tau_i, \mathsf{info}_i)$ and for each query receives a valid descriptor $(A_{\tau_i}, E_{k_{\tau_i}}(\mathsf{info}_i), \sigma_i)$ for epoch τ_i and identity key A.

3. $\mathcal{A}$ may at any time make a single *challenge* query $(\tau^*, \mathsf{info}_0^*, \mathsf{info}_1^*)$, as long as $\tau^* \neq \tau_i$ for any $i \in \{1, \ldots, q_E\}$. In response, a bit b is chosen uniformly at random. If $b = 0$, $\mathcal{A}$ receives as a response a valid descriptor $(A_{\tau^*}, E_{k_{\tau^*}}(\mathsf{info}_0^*), \sigma^*)$ for epoch τ^* and identity key A. Otherwise $\mathcal{A}$ receives as a response (A^*, c^*, σ^*), where A^* is a randomly generated identity key, $c^* = E_{k^*}(\mathsf{info}_1^*)$ for a randomly chosen AES key k^*, and σ^* is a valid signature on c^* under identity key A^*.

4. After expending t computational resources and making at most q_H hash oracle queries, $\mathcal{A}$ halts with an output b^*.

5. If $b^* = b$, the experiment has output 1, otherwise the experiment has output 0.

We say that a scheme is $(t, q_E, q_H, \epsilon_{\mathsf{link}})$-unlinkable if for every t-bounded $\mathcal{A}$, $\Pr[\mathsf{Link}_{\mathcal{A}}(t, q_E, q_H) = 1] - \frac{1}{2} \leq \epsilon_{\mathsf{link}}$.

4.2 Unforgeability

We define the (t, q_D, q_H)-descriptor forgery experiment with adversary $\mathcal{A}$, $\mathsf{DForge}_{\mathcal{A}}(t, q_D, q_H)$ as follows:

1. An identity key $A = aB$ and secret k are generated according to the description in section 3. A is given to the adversary $\mathcal{A}$.

2. $\mathcal{A}$ makes q_D adaptive "descriptor" queries (τ_i, c_i) and for each query receives a valid descriptor $(A_{\tau_i}, c_i, \sigma_i)$ for epoch τ_i and identity key A.

3. After expending at most t computational resources and making at most q_H hash oracle queries, $\mathcal{A}$ halts with an output (A^*, c^*, σ^*).

4. If there exists a τ such that $A^* = A_\tau$, (τ, c^*) was not a query of $\mathcal{A}$, and σ^* is a valid signature on c^* under epoch identity A^*, the output of the experiment is 1, otherwise it is 0.

We say that a scheme is $(t, q_D, q_H, \epsilon_{\mathsf{forge}})$-descriptor unforgeable if for every t-bounded $\mathcal{A}$,$\Pr[\mathsf{DForge}_{\mathcal{A}}(t, q_D, q_H)] \leq \epsilon_{\mathsf{forge}}$.

5 Security Proofs

5.1 Unlinkability

The proof of unlinkability is fairly straightforward: if an adversary never queries the hash oracles with the identity key A, then all of the epoch identities will look independent of A, and thus the probability of querying at A will be negligible.

Theorem 1. *If all hash functions are implemented as random oracles, and AES is* $(t, \epsilon_{\mathsf{pre}})$*-secure, then Ransom's scheme is* $(t, q_E, q_H, \epsilon_{\mathsf{link}})$*-unlinkable, where* $\epsilon_{\mathsf{link}} \leq \frac{q_H}{\ell - q_H} + \epsilon_{\mathsf{pre}}$.

Proof. We construct a pair of hybrid experiments, $\mathsf{Game}_{0,\mathcal{A}}$ and $\mathsf{Game}_{1,\mathcal{A}}$, such that for any attacker $\mathcal{A}$, we will have:

$$|\Pr[\mathsf{Link}_{\mathcal{A}} = 1] - \Pr[\mathsf{Game}_{0,\mathcal{A}} = 1]| \leq \tfrac{q_H}{\ell - q_H} \quad (1)$$

$$|\Pr[\mathsf{Game}_{0,\mathcal{A}} = 1] - \Pr[\mathsf{Game}_{1,\mathcal{A}} = 1]| \leq \epsilon_{\mathsf{pre}} \quad (2)$$

$$\Pr[\mathsf{Game}_{1,\mathcal{A}} = 1] = \tfrac{1}{2} \quad (3)$$

By the triangle inequality, this will imply the theorem.

Experiment $\mathsf{Game}_{0,\mathcal{A}}$**.** We define $\mathsf{Game}_{0,\mathcal{A}}$ as follows:

1. An identity key $A = aB$ and secret k are generated according to the description in section 3.

2. $\mathcal{A}$ makes q_E adaptive queries $(\tau_i, \mathsf{info}_i)$ and for each query a uniform identity key A_{τ_i} and secret key k_i are chosen (independently of A and k) and used to generate a valid descriptor $(A_{\tau_i}, E_{k_i}(\mathsf{info}_i), \sigma_i)$.

3. $\mathcal{A}$ may at any time make a single *challenge* query $(\tau^*, \mathsf{info}_0^*, \mathsf{info}_1^*)$, as long as $\tau^* \neq \tau_i$ for any $i \in \{1, \ldots, q_E\}$. In response, a bit b, identity key A^* and symmetric key k^* are chosen uniformly at random. $\mathcal{A}$ receives in response (A^*, c^*, σ^*), where $c^* = E_{k^*}(\mathsf{info}_b^*)$ and σ^* is a valid signature on c^* under identity key A^*.

4. If at any time $\mathcal{A}$ makes a hash oracle query of the form $H_e(t||A)$ or $H_d(t||A)$, the experiment immediately halts with output 0. Otherwise the game continues until $\mathcal{A}$ outputs a bit b^*.

5. If $b^* = b$, the experiment has output 1, otherwise the experiment has output 0.

To prove inequality (1) we define a series of events over all three experiments. We let Q_i denote the event that the i-th hash oracle query has the form $H_e(t||A)$ or $H_d(t||A)$. We define $\mathsf{F}_i = \bigvee_{j \leq i} \mathsf{Q}_j$, and $\mathsf{F} = \mathsf{F}_{q_H}$. The inequality then follows from two observations.

First, conditioned on $\neg\mathsf{F}_i$, the view of $\mathcal{A}$ up through the i-th hash oracle query is identically distributed in both experiments: if the view of $\mathcal{A}$ does not include s_{τ_j} then $s_{\tau_j} A$ (in $\mathsf{Link}_{\mathcal{A}}$) and A_{τ_j} (in $\mathsf{Game}_{0,\mathcal{A}}$) are both identically and independently distributed, and likewise for k_{τ_j} and k_j. So

$$\Pr[\mathsf{Link}_{\mathcal{A}} = 1 | \neg\mathsf{F}] = \Pr[\mathsf{Game}_{0,\mathcal{A}} = 1 | \neg\mathsf{F}] \ .$$

Second, in either experiment we therefore have $\Pr[\mathsf{Q_i} | \neg\mathsf{F}_{i-1}] \leq \frac{1}{\ell - i}$, since conditioned on $\neg\mathsf{F}_{i-1}$ the view of $\mathcal{A}$ is independent of A. Thus $\Pr[\mathsf{F}_i] \leq \sum_{j \leq i} \frac{1}{\ell - j}$, and

$$\Pr[\mathsf{F}] \leq \frac{q_H}{\ell - q_H} \ .$$

Experiment $\mathsf{Game}_{1,\mathcal{A}}$**.** The experiment $\mathsf{Game}_{1,\mathcal{A}}$ is defined identically to the experiment $\mathsf{Game}_{0,\mathcal{A}}$, with the exception that the ciphertext c^* returned for the challenge query is replaced by $|\mathsf{info}| + 128$ uniform random bits. In this experiment it is clear that the view of $\mathcal{A}$ is independent of b, and therefore $\Pr[b^* = b] = \frac{1}{2}$. But distiguishing between these experiments is exactly the task of distinguishing AES encryptions from uniform random bits, so for any $\mathcal{A}$ we have

$$|\Pr[\mathsf{Game}_{0,\mathcal{A}} = 1] - \Pr[\mathsf{Game}_{1,\mathcal{A}} = 1]| \leq \epsilon_{\mathsf{pre}} \ .$$

□

5.2 Unforgeability

The proof of descriptor unforgeability is a mostly straightforward reduction to forging Ed25519 signatures, with a factor of T security loss. The reduction works by guessing which epoch t^* the descriptor forgery will cover, "planting" the Ed25519 public key A as A_{t^*} by choosing a uniform $s_{t^*} \bmod \ell$, and setting the identity key to $A' = s_{t^*}^{-1} A$. From there we exploit oracle separation to respond to H_m queries on values $(A^\dagger, s)$ where $A^\dagger \neq A$ with values that allow us to forge signatures for these identity keys.

Theorem 2. *If all hash functions are implemented as random oracles, and Ed25519 signatures are* $(t, q_S, \epsilon_{\mathsf{sig}})$*-secure against existential forgery, then Ransom's scheme is* $(t, q_D, q_H, \epsilon_{\mathsf{forge}})$*-secure against descriptor forgery, where* $q_D \leq q_S$ *and*

$$\epsilon_{\mathsf{forge}} \leq T\epsilon_{\mathsf{sig}} + \frac{q_D q_H}{\ell} + q_H 2^{-b} \ .$$

Proof. For any attacker $\mathcal{A}$, we define two hybrid experiments $\mathsf{Game}_{0,\mathcal{A}}$ and $\mathsf{Game}_{1,\mathcal{A}}$, and an algorithm $\mathcal{F}_{\mathcal{A}}$ with the same complexity as $\mathcal{A}$ so that:

$$|\Pr[\mathsf{DForge}_{\mathcal{A}} = 1] - \Pr[\mathsf{Game}_{0,\mathcal{A}} = 1]| \leq q_H 2^{-b} \tag{4}$$

$$|\Pr[\mathsf{Game}_{0,\mathcal{A}} = 1] - \Pr[\mathsf{Game}_{1,\mathcal{A}} = 1]| \leq \tfrac{q_D q_H}{\ell} \tag{5}$$

$$\Pr[\mathsf{SForge}_{\mathcal{F}_{\mathcal{A}}} = 1] \geq \tfrac{1}{T} \Pr[\mathsf{Game}_{1,\mathcal{A}} = 1] \tag{6}$$

The theorem follows from the conjunction of these inequalities.

Experiment $\mathsf{Game}_{0,\mathcal{A}}$**.** In this experiment, we run the descriptor forgery experiment, with three modifications:

- $\mathcal{A}$ is modified so that it never makes the same query (τ, c) twice. Instead, it checks before queries and reuses the earlier result if a duplicate is detected.
- All Ed25519 signatures replace the step $r = H_r(k, t, m)$ by choosing $r \in_R \{0, \ldots, \ell - 1\}$.
- If there is ever a hash oracle query of the form $H_r(k, t, m)$ using the k value in the signing key, the output of the experiment is set to 0, regardless of the outcome of the forgery attempt.

Note that similarly to the previous proof, until $\mathcal{A}$ queries H_r at one of the points used in some signature, its view is identical in either world, and indpendent of k. Thus the probability, in q_H queries of finding the b-bit value k is at most $q_H 2^{-b}$.

Experiment $\mathsf{Game}_{1,\mathcal{A}}$. This experiment further modifies $\mathsf{Game}_{0,\mathcal{A}}$ as follows: when $\mathcal{A}$ makes a descriptor query (τ, c), the descriptor (A_τ, c, σ) is computed as in $\mathsf{Game}_{0,\mathcal{A}}$. The signature σ is then parsed as a pair $(\underline{R}, s)$, and if $\mathcal{A}$ has previously queried $H_m(\underline{R}, A_\tau, c)$, the outcome of the experiment is set to 0, regardless of the outcome of the forgery event.

Let P_i denote the event that the i-th descriptor query results in an outcome of 0, and $\mathsf{P} = \bigvee_{i \leq q_D} \mathsf{P}_i$. Then since nothing in the view of $\mathcal{A}$ changes between the experiments, we have $|\Pr[\mathsf{Game}_{1,\mathcal{A}} = 1] - \Pr[\mathsf{Game}_{0,\mathcal{A}} = 1]| \leq \Pr[\mathsf{P}]$. Since the value $R = rB$ is chosen uniformly at random in each descriptor query and $\mathcal{A}$ makes at most q_H queries to H_m, we have $\Pr[\mathsf{P}_i] \leq \frac{q_H}{\ell}$, and thus by the union bound $\Pr[\mathsf{P}] \leq \frac{q_H q_D}{\ell}$.

Algorithm $\mathcal{F}_\mathcal{A}$. We can now construct an Ed25519 forger from $\mathcal{A}$ as follows. $\mathcal{F}_\mathcal{A}$ is given as input an Ed25519 key verification key A, has oracle access to the hash functions H_m, and will simulate the hash functions H_e, H_d for $\mathcal{A}$. (It will also simulate some of the queries to H_m) $\mathcal{F}_\mathcal{A}$ runs as follows:

- Given verification key A, choose a "target epoch" $t \in_R \{0, \ldots, T\}$, choose a random value $s_t \in_R \{0,1\}^{2b}$, and set $s = s_t^{-1} \bmod \ell$. Set $A* = sA$ and record $H_e(t||A*) = s_t$. Run $\mathcal{A}$ with A^* as the target identity key.

- When $\mathcal{A}$ makes hash query of the form $H_e(\tau||A^*)$, $\mathcal{F}_\mathcal{A}$ checks whether it has recorded a value s_τ for the query; if not, it chooses a value $s_\tau \in_R \{0,1\}^{2b}$ and records this value. s_τ is returned.

- When $\mathcal{A}$ makes a hash query of the form $H_m(\underline{R}, A', m)$, $\mathcal{F}_\mathcal{A}$ first checks whether $A' = A$, and if so queries its H_m oracle for the correct value. Otherwise, $\mathcal{F}_\mathcal{A}$ checks whether it has previously recorded a value h for $(\underline{R}, A', m)$; if not, a value $h \in_R \{0,1\}^{2b}$ is chosen and recorded. The recorded value h is returned.

- When $\mathcal{A}$ makes a descriptor query (τ, c), $\mathcal{F}_\mathcal{A}$ checks whether $\tau = t$; if so, $\mathcal{F}_\mathcal{A}$ queries its signing oracle for a signature on the message c and returns this value. Otherwise, $\mathcal{F}_\mathcal{A}$ simulates a query to $H_e(\tau||A^*)$ to get s_τ and $A_\tau = s_\tau A^*$. Next, $\mathcal{F}_\mathcal{A}$ chooses $h \in_R \{0,1\}^{2b}$, $\sigma \in_R \{0, \ldots, \ell - 1\}$, and computes $R = sB - hA_\tau$. If $\mathcal{A}$ has previously made a hash query of the form $H_m(\underline{R}, A_\tau, c)$, $\mathcal{F}_\mathcal{A}$ halts with output Fail. Otherwise, $\mathcal{F}_\mathcal{A}$ records $H_m(\underline{R}, A_\tau, c) = h$ and returns the descriptor (A_τ, c, σ), where $\sigma = (\underline{R}, s)$.

- When $\mathcal{A}$ outputs a descriptor forgery (A', c, σ), check if $A' = A$. If $A' = A$ then output (c, σ) as the signature forgery, otherwise stop with output Fail.

Let us denote by SF the event that $\mathcal{A}$'s output (A', c, σ) is a valid descriptor for some A_τ. Then, since $\mathcal{A}$'s view in this simulation is identically distributed to the experiment

$\mathsf{Game}_{1,\mathcal{A}}$, we have $\Pr[\mathsf{SF}] \geq \Pr[\mathsf{Game}_{1,\mathcal{A}} = 1]$. [1] Furthermore, we see that $\Pr[\mathsf{SForge}_{\mathcal{F}_\mathcal{A}} = 1] = \Pr[\mathsf{SF} \wedge A' = A]$. Since the view of $\mathcal{A}$ is independent of the target epoch t, due to the uniform choice of $A \in_R \langle B \rangle$ and the independence of the A_τ, we have $\Pr[A' = A] = \frac{1}{T}$ and thus

$$\Pr[\mathsf{SForge}_{\mathcal{F}_\mathcal{A}} = 1] = \frac{1}{T} \Pr[\mathsf{Game}_{1,\mathcal{A}} = 1] \ .$$

□

6 Modifications to Ransom's scheme in Proposal 224

Proposal 224 modifies the scheme as described here in several ways; we summarize each modification and its effect (if any) on the security bounds above:

- **Ephemeral Signing Keys:** in Proposal 224, rather than using the epoch signing key to sign the descriptor directly, an ephemeral (independent, uniformly random) Ed25519 key pair is chosen for each epoch. The ephemeral verification key is signed by the epoch signing key, and then the ephemeral signing key is used to sign the (encrypted) descriptor plus the certification:

  This change allows the identity key to be stored offline, and has no effect on the security levels for either the unlinkability or descriptor forgery game, since in the unlinkability game, the ephemeral keys are independent of the epoch keys and in the descriptor forgery game, the adversary is allowed to submit arbitrary info strings for signing under epoch keys, which could be ephemeral verification keys.

- **The hash function** H_d In proposal 224, the hash function H_d is implemented roughly as $H_d(t||A) = H(A_t||H(\text{"}\texttt{subcredential}\text{"}||A||A_t)||\text{"}\texttt{hsdir}-\texttt{encrypted}-\texttt{data}\text{"})$. Since this still involves querying H at the identity key A in a parseable location, this does not alter the security bound in any way.

- **Epoch Length and** H_e Proposal 224 specifies epochs of length 25 hours but encodes the epoch number as a 64-bit integer. If we interpret the value T literally this leads to a factor of 2^{64} security loss. However, since there is no advantage to forging a descriptor for a time period after we're all dead, we can restrict the security game to consider a maximum time period $T \ll 2^{64}$, for example $T = 2^{16}$ leads to a 16-bit security loss and is realistic as long as the deployment of the present scheme is expected to last less than 144 years.

- **Hashing prefixes** Proposal 224 also adapts a different set of unique prefixes to separate the remaining hash functions. This has no effect whatsoever on the security of the resulting scheme, since the only important characteristic of these prefixes is their distinctness.

[1] The simulation may fail if $\mathcal{A}$ manages to query H_k at some point related to the secret k chosen by the Ed25519 challenger; until such a query the simulation is identical to $\mathsf{Game}_{1,\mathcal{A}}$, and conditioned on the query, we know that $\mathsf{Game}_{1,\mathcal{A}} = 0$, while the event SF may still occur

References

[1] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012.

[2] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, May 2013.

[3] Robert Ransom. Fwd: New HS protocol. Email to tor-dev list: `https://lists.torproject.org/pipermail/tor-dev/2012-September/004026.html`, September 2012.