Verified Multiple-Time Signature Scheme from One-Time Signatures and Timestamping

Denis Firsov, Henri Lakk, Ahto Truu
{denis.firsov, henri.lakk, ahto.truu}@guardtime.com

GuardTime

June 10, 2021

Background

- Due to the quantum threat hash-based signatures gain interest again (e.g., SPHINCS, XMSS).
- Efficiency (time and space) is a concern for hash-based signatures.
- Buldas, Laanoja, and Truu proposed a new type of server-assisted hash-based signature schemes (BLT schemes) relying on the security properties of the timestamping (2017).
- Unfortunately, original BLT signatures have either expensive key-generation phase or stateful client-side computations. Moreover, security proofs were performed in one-time use setting.

Our goals

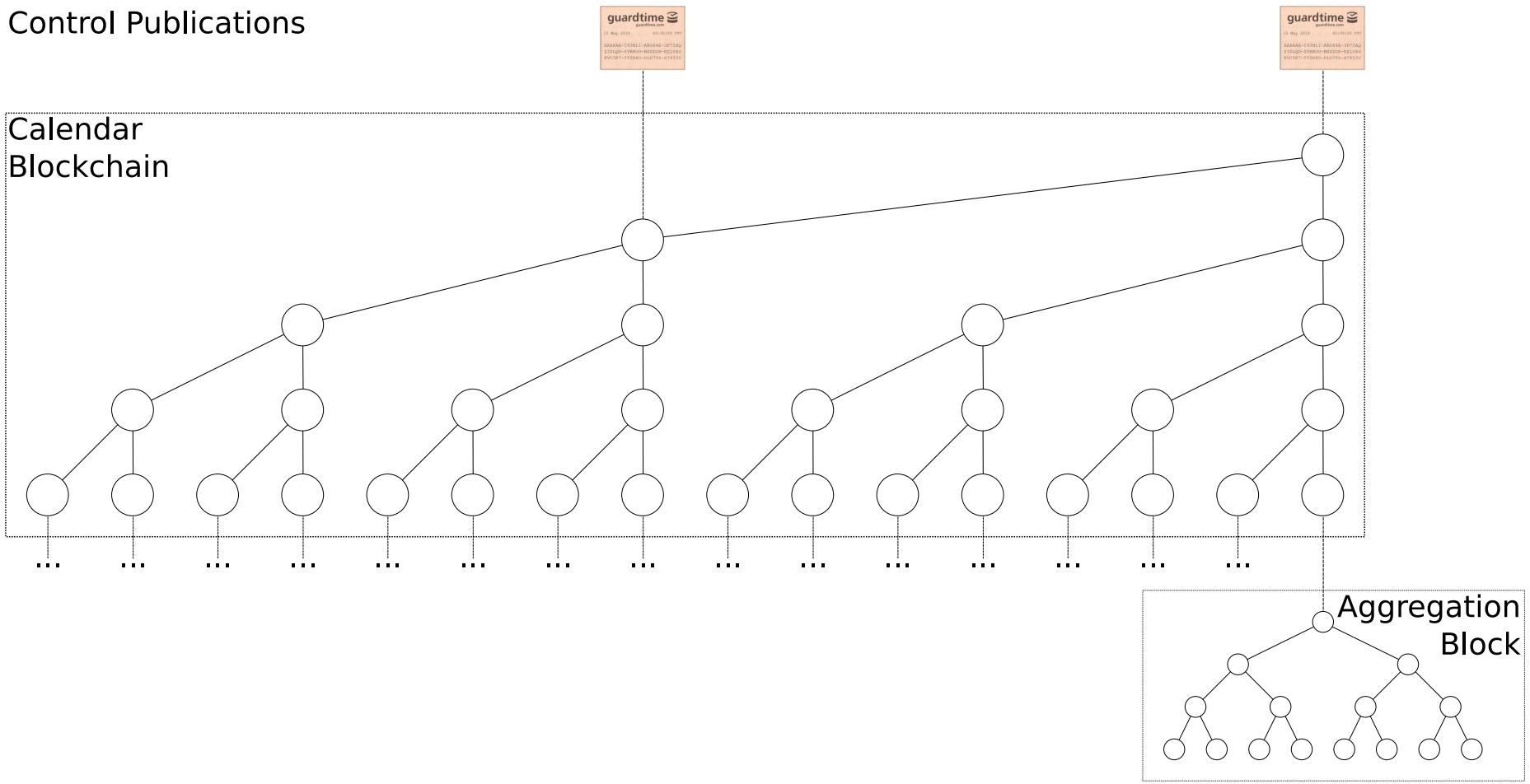
- Extend BLT signature to have efficient and stateless client-side computations.
- Formally prove existential unforgeability in the multiple-time setting.

Cryptographic Timestamping

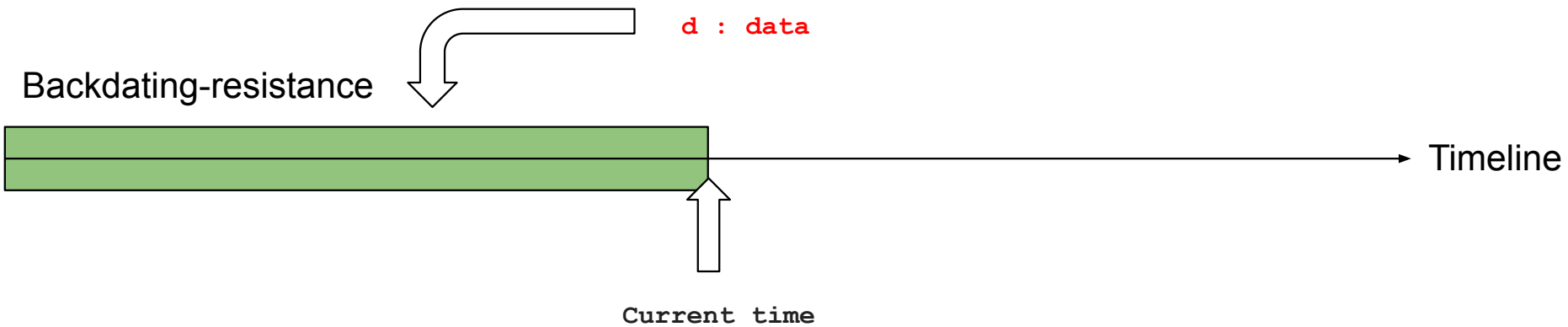
1. Cryptographic timestamping generates proofs that data existed before a particular time.
2. Common approach is to use a trusted third party (e.g., notary).
3. Haber and Stornetta made the first steps toward trustless timestamping by proposing a scheme where each timestamp would include some information from the immediately preceding one and a reference to the immediately succeeding one.

Cryptographic Timestamping

Contemporary timestamping services (e.g., KSI Blockchain Timestamping) aggregate the user queries into Merkle trees and publish their roots in widely accessible media (e.g., newspapaers).



Cryptographic Timestamping: Backdating resistance



Cryptographic Timestamping: Ideal model

```
1:  module X
2:      var T : ℕ
3:      var m : (time, data)map
4:
5:      fun timestamp(t : time, d : data) = {
6:          if T < t then
7:              T ← t
8:              m[T] ← d
9:          end if
10:     }
11:
12:     fun check(t : time, d : data) = {
13:         return m[t] = d
14:     }
15: end
```

Definition (Tag Systems)

A *tag system* is a triple of probabilistic polynomial-time (PPT) algorithms $(G_{\mathcal{T}}, T_{\mathcal{T}}, V_{\mathcal{T}})$, where $G_{\mathcal{T}}$ is a key pair generation algorithm, $T_{\mathcal{T}}$ a tagging algorithm, and $V_{\mathcal{T}}$ a tag verification algorithm.
The tag system is *correct* if the algorithm $V_{\mathcal{T}}$ agrees with $T_{\mathcal{T}}$ for all valid key pairs and messages:

$$\Pr\left[(pk, sk) \leftarrow G_{\mathcal{T}} : V_{\mathcal{T}}(pk, T_{\mathcal{T}}(sk, t), t) = 1\right] = 1.$$

Tag Systems: Forward Resistance

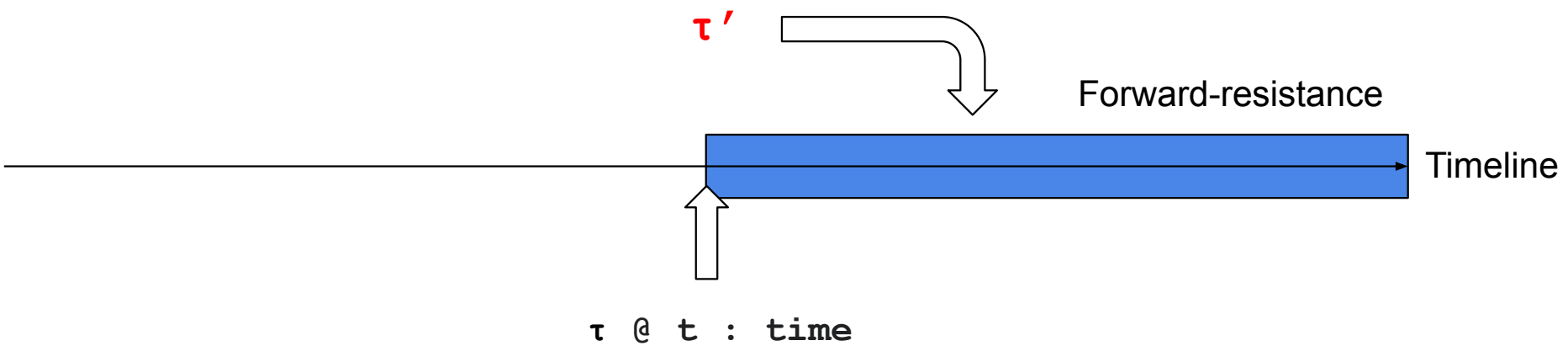
Definition

The tag system is *forward resistant* if the probability

$$\Pr\left[\begin{array}{c}(pk, sk) \leftarrow G_{\mathcal{T}},\ (\tau, t) \leftarrow A^{T_{\mathcal{T}}(sk,\cdot)}(pk) : \\ V_{\mathcal{T}}(pk, \tau, t) = 1,\ t_{max} < t\end{array}\right],$$

$$\text{where } t_{max} = max\ T_{\mathcal{T}}.log$$

is negligible for any PPT adversary A.



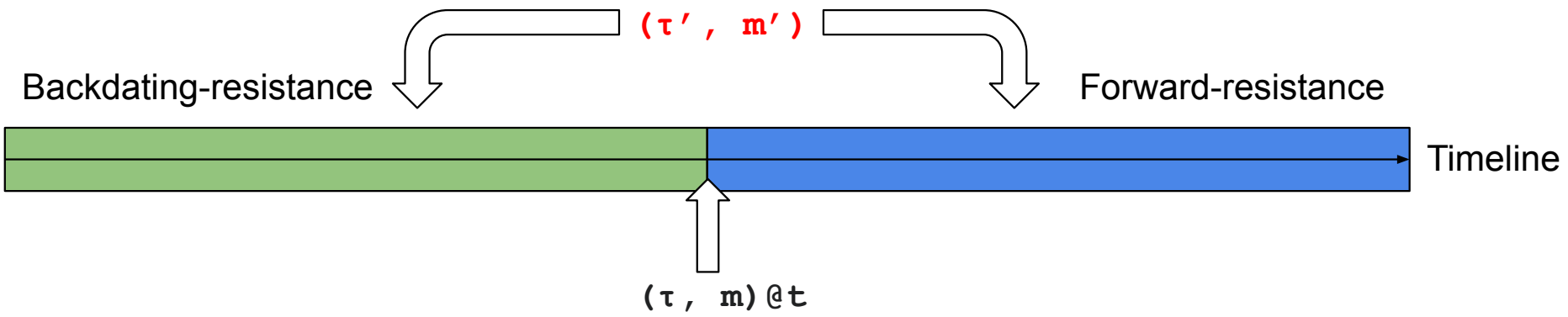
BLT Scheme = Timestamping Service + Tag System

- Generate a public-private keypair $(pk, sk) \leftarrow G_{\mathcal{T}}$ of the tag system.
- To sign a message m:
  1. Query the time t of the timestamping service $\mathcal{X}$.
  2. Use the private key to generate a tag τ for the "next" time $t+1$;.
  3. Bind the tag with the message by timestamping the pair (τ, m).
  4. Output $(\tau, t+1)$ as a signature of m.
- To verify m against signature (τ, t):
  1. Verify τ against the time t and the public key pk.
  2. Verify that the timestamping service contains a binding of τ and m at time t.

BLT Signature Scheme: Existential Unforgeability

Theorem

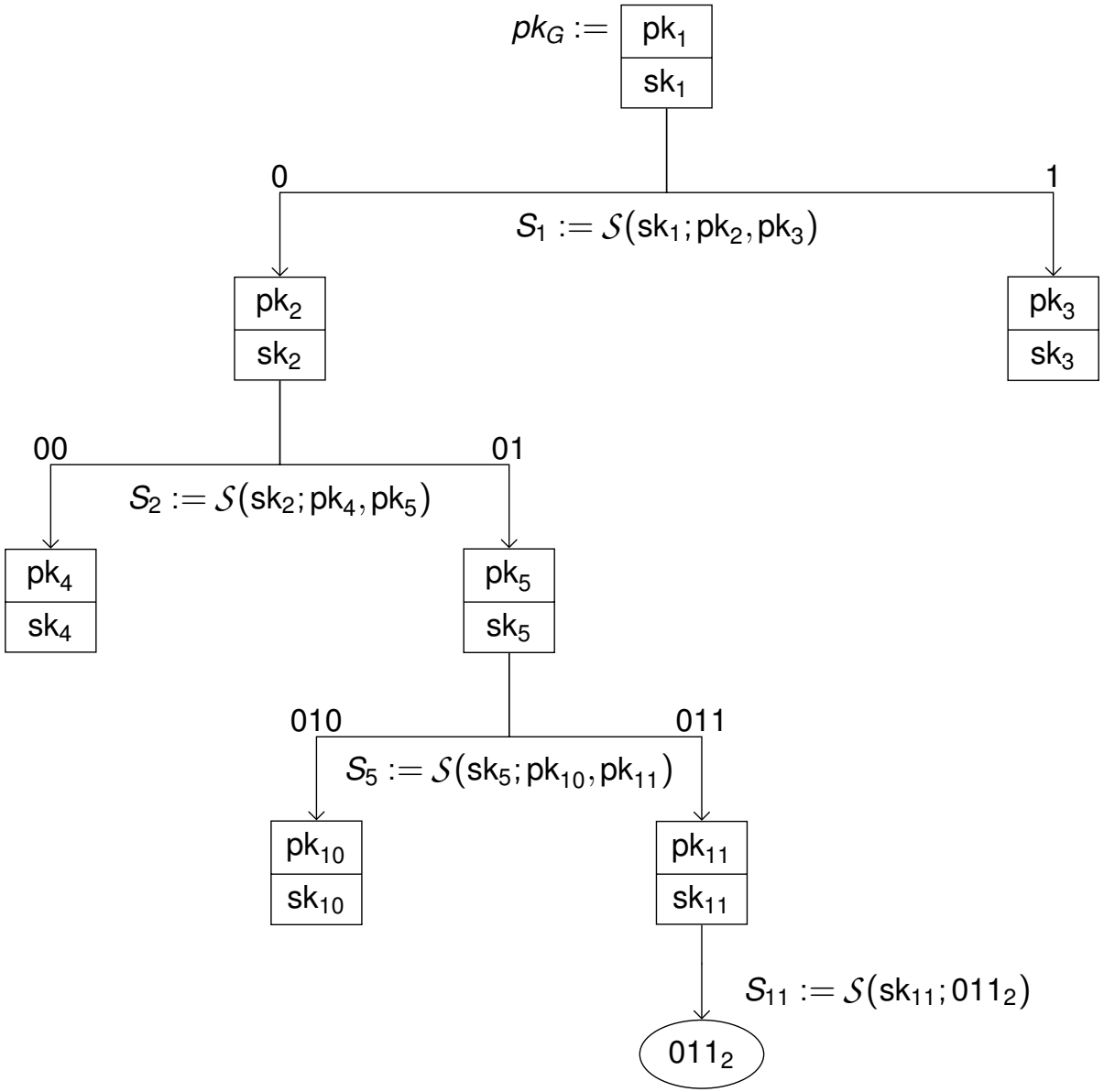
If the honest backdating resistant timestamping repository holds plain message-tag pairs then the probability of an adversary performing a successful signature forgery is bounded from above by the probability of breaking the forward-resistance of a tag system.



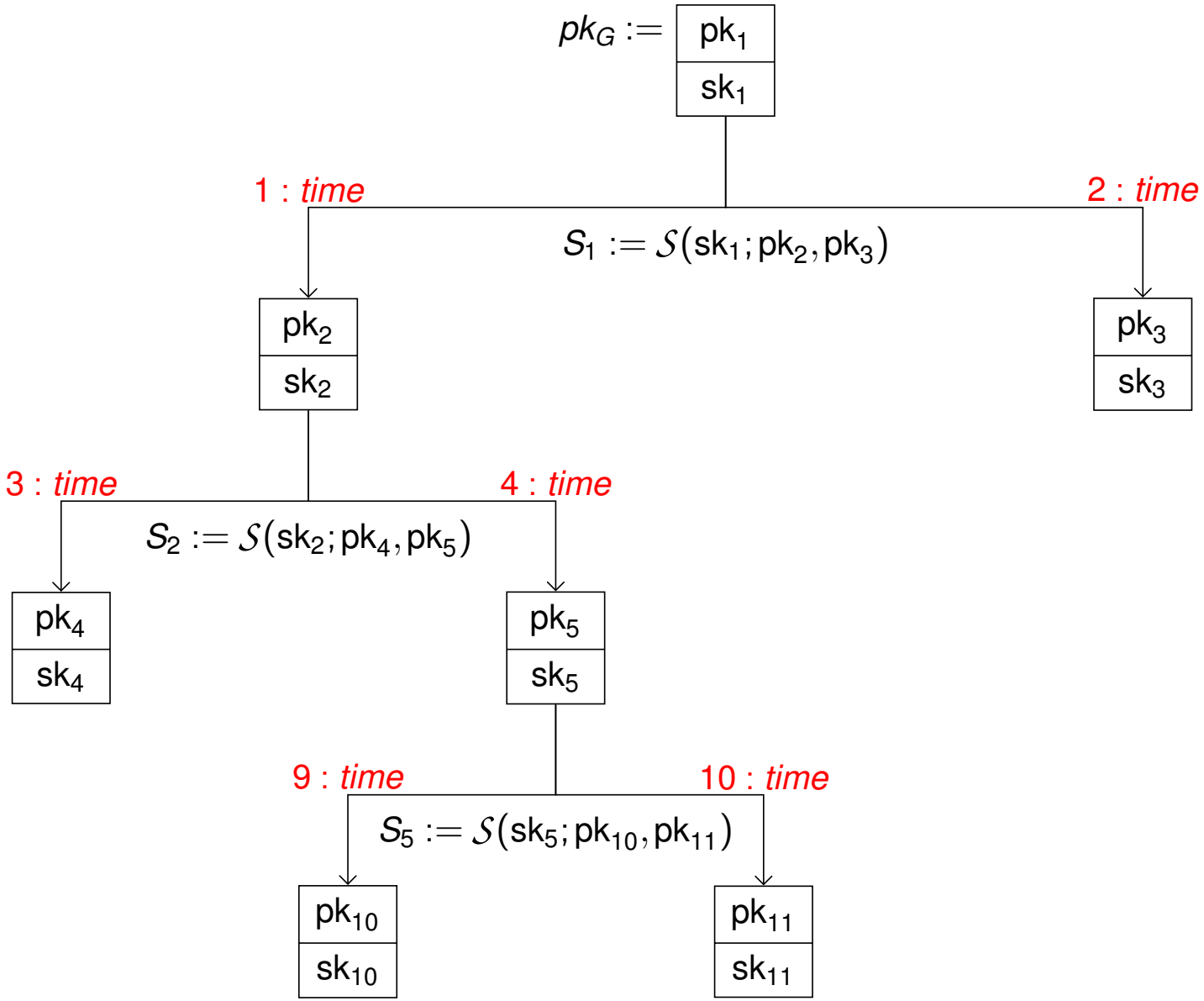
Next goal

To instantiate BLT Signature Scheme we need to construct a multiple-time forward-resistant tag system.

Goldreich Signature Scheme



Many-Time Tag-System



Theorem

The probability of breaking the forward resistance of multiple-time tag system with N pre-generated independent secret keys is bounded from above by $N \cdot E$, where E is the probability of breaking the existential unforgeability of the one-time signature scheme.

- Pregenerating one-time keys for the whole period of lifetime of BLT keypair is inefficient.
- However, the presented datastructure allows lazy key-evaluation from a PRF.
- Deriving the one-time keypairs from a PRF drops the security level by indistinguishability of the PRF.

EasyCrypt: Formalization

- Our definitions, constructions, correctness, and security claims are fully formalized in EasyCrypt framework.
- Our formalization only required standard techniques:
  - Adversaries modelled as computations satisfying abstract interfaces.
  - Probability claims are relative to each other. As a result, must manually verify the complexity of transformations.
  - Security protocols are modelled by imperative programs.
  - Proofs are done using the standard game-rewriting technique via Probabilistic Relational Hoare Logic.
- Due to the support of SMT solvers, proving pure mathematical statements as well as properties of the functional programs and datastructures is easy. Moreover, proofs also survive (small) changes in the implementation.
- The hardest and most fragile are proofs about distributions (e.g., key generation) which are defined as imperative programs.

Thank you!