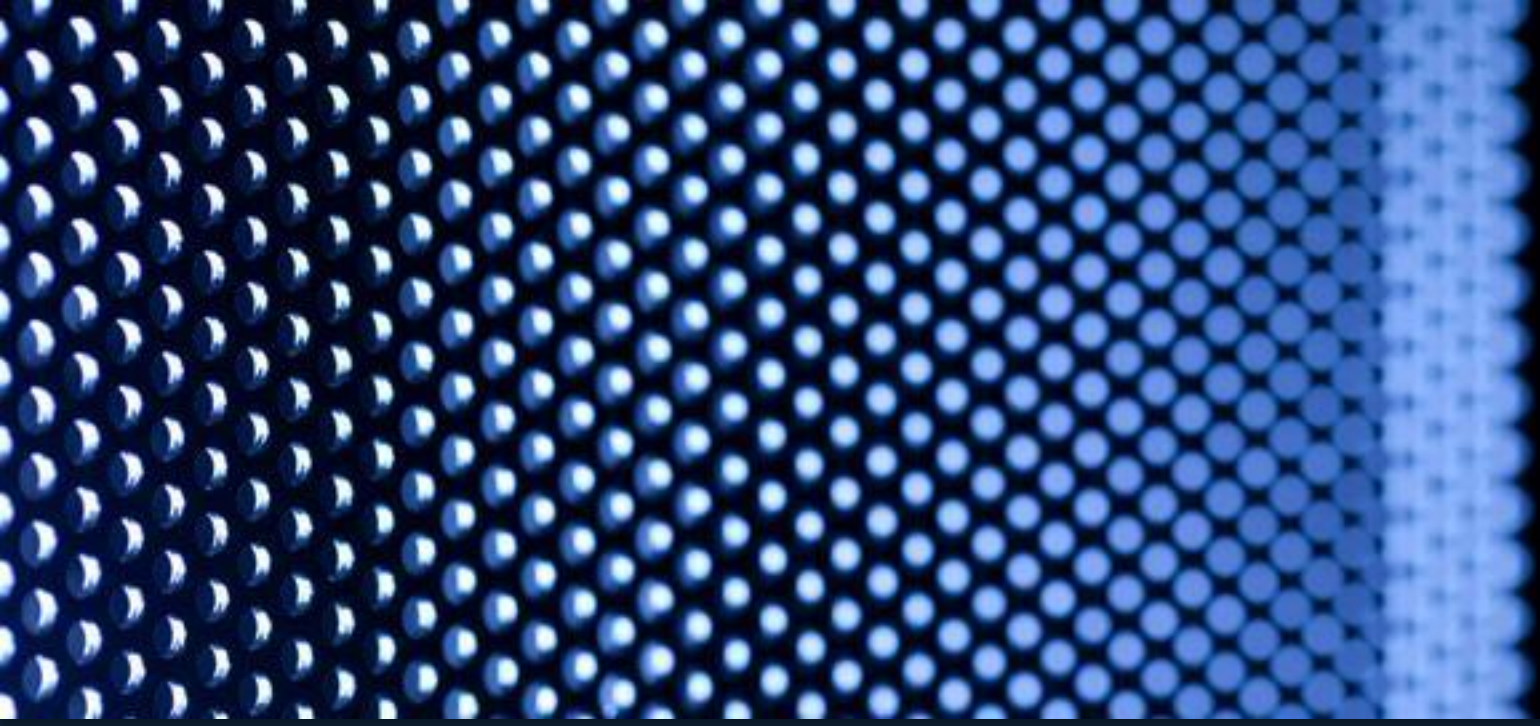


VERISIGN

Rethinking Adoption of Hash Signatures

Dr. Burt Kaliski, Jr.

Senior Vice President, Chief Technology Officer

ETSI 2nd Quantum-Safe Cryptography Workshop

October 6, 2014

Hash Signatures

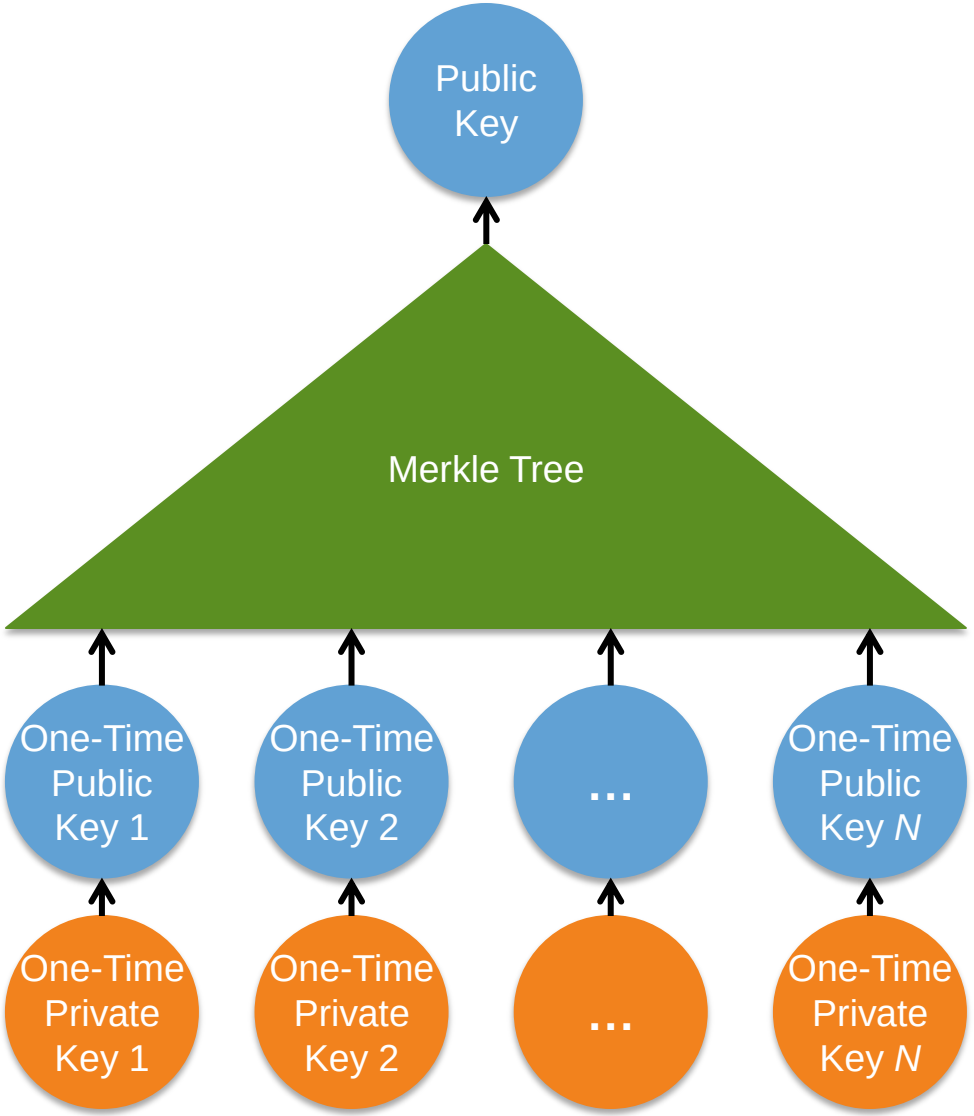
Hash Signatures: Background

- For purposes of this discussion, “hash signature” = **Merkle Tree Signature** with one-time signature scheme based on hash function (e.g., Lamport-Diffie-Winternitz)
- References:
  - [MC14] D. McGrew and M. Curcio. *Hash-Based Signatures*. Internet-Draft draft-mcgrew-hash-sigs-02, July 4, 2014.
  - [BDH11] J. Buchmann, E. Dahmen and A. Hülsing. XMSS – A Practical Forward Secure Signature Scheme based on Minimal Security Assumptions. *PQCrypto 2011*.
- “Conventional signature” = RSA, ECDSA, etc.
- Assumption: Hash signatures “quantum safe” as a general construction (with appropriate parameter sizes)
  - May need to replace hash function over time, but easier to develop new hash function than entirely new signature scheme!

Hash Signatures: General Model

Signature includes:

- **One-time signature** with one-time private key
- **Index** of one-time key pair
- **Authentication path** from one-time public key to root



Key Question: Driving Adoption

- Assuming that hash signatures are better in the long term, what do we need to encourage adoption?
- Challenge: Long-term advantages generally aren't enough

Long-Term Advantages Aren't Enough

- Historically, crypto algorithm adoption has been motivated by three factors: **mandates**, **algorithm breaks**, and **significant new functionality**
  - Similar point for key size increases
- Partial breaks often patched
- Potential future breaks (e.g., via quantum computers, or advances in cryptanalysis) generally ignored
- Premise: Long-term advantages of hash signatures, other quantum-safe crypto **not enough to motivate adoption**
  - Even in new applications where interoperability isn't as important …
  - Economic tradeoff: If it's not broken – fix something else!
- Without mandates or breaks in other algorithms, also need **near-term advantages**: new functionality

What Kinds of New Functionality?

- Primarily, improvements in **trust model** – which parties have to be trusted, for what purposes, and for how long
- "Shorter" and "faster" help but only they change the game – e.g., by making other functionality practical sooner
  - Moore's Law, hardware accelerators, hybrid algorithms, etc. quickly level the playing field

- Public-key cryptography enabled **encryption, authentication without prior establishment of shared secrets** – advantage even though size, speed were not!
- Elliptic curve cryptography shorter, faster than RSA for most operations – but especially for key generation, which enables **forward secrecy**

A Health Care Analogy

- "… despite decades of effort and millions of dollars, only between 3% and 34% of people in poor countries regularly wash their hands …"
- "The bigger problem is that **long-term health considerations do not drive behavior** …"
- "What does are things like love, fear, and wanting to be accepted and admired"

Source: It's a wash: Hands-on hygiene in Peru. *Science,* 12 September 2014.

Some Near-Term Advantages of Hash Signatures

#1: Short Backdating Windows

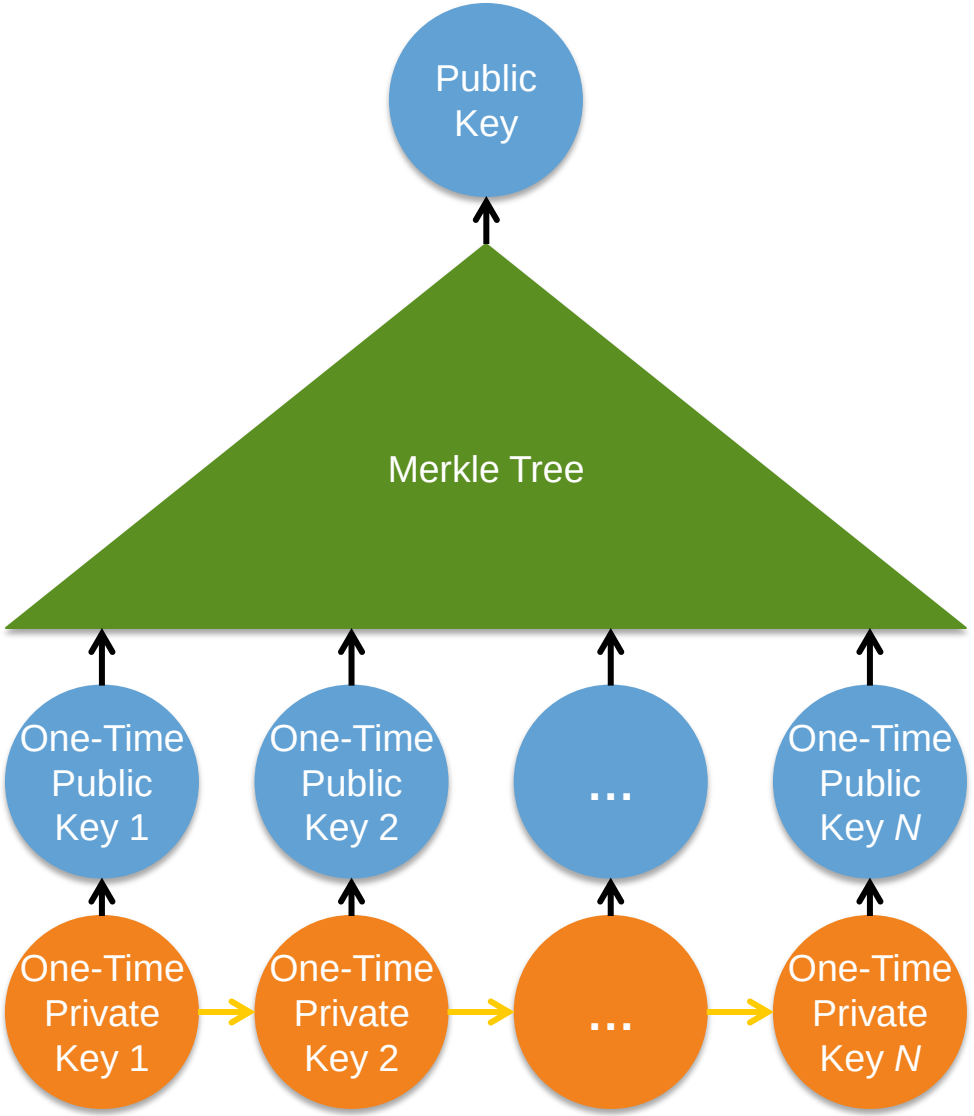
- With conventional signatures, adversary who compromises private key can **backdate signatures** all the way to start of **validity period** for public key
  - e.g., as published in certificate
- With hash signatures, and [BDH11] “forward-secure” enhancement, adversary can only backdate to start of validity period for **current one-time private key**

- Advantage: Short backdating windows **without frequent key rollovers** / certificate updates
  - Trust model improvement: Signer can **bound impact of private key compromise** to shorter time period
- Application: Time-based transaction signing

Short Backdating Windows

Forward-secure enhancement (based on [BDH11]):

- Generate next one-time private key as **one-way function** of previous one
  - … or of related state
- Associate indices with specific **sub-intervals** of overall validity period



#2: Coarse-Grained Delegation

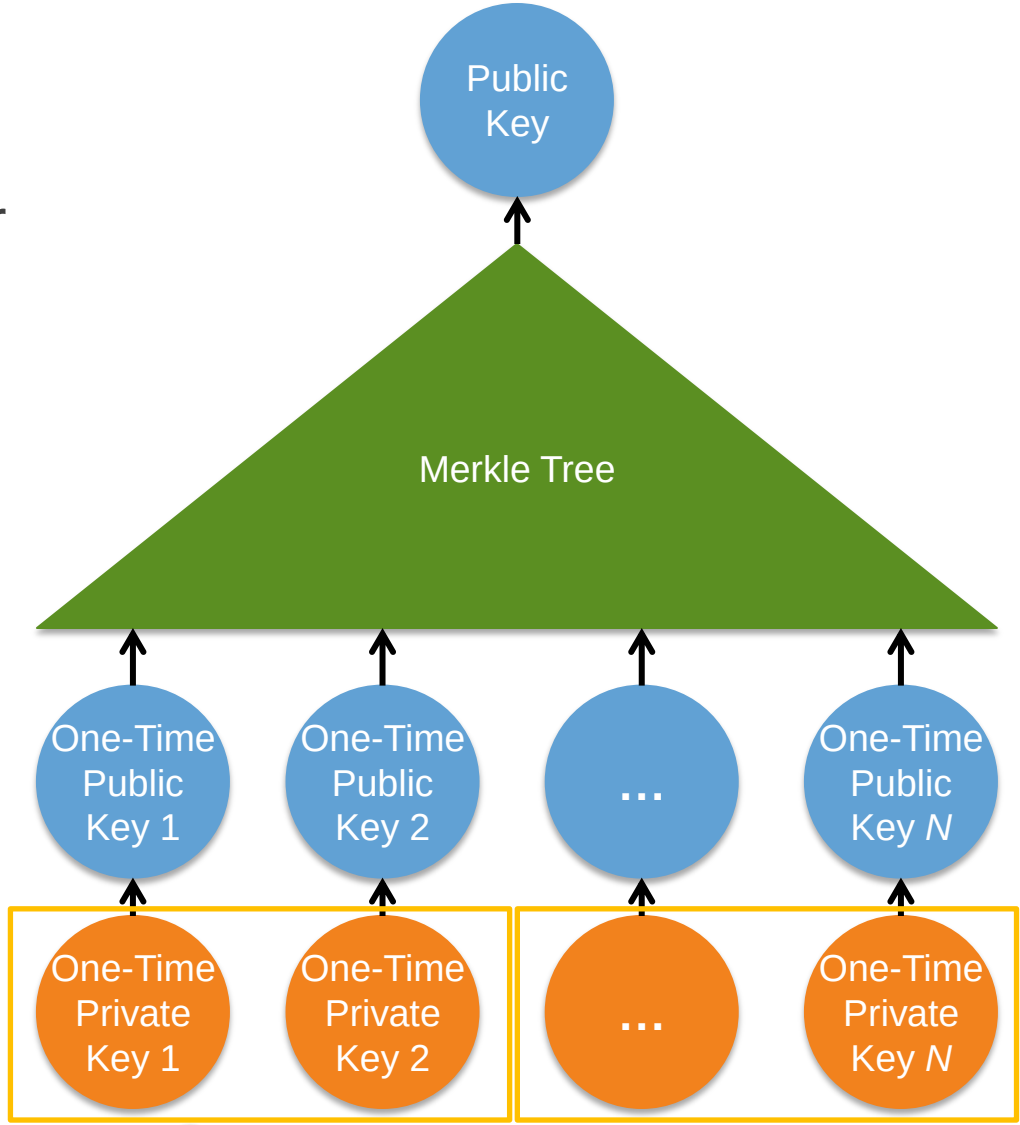
- With conventional signatures, signer can only delegate limited signing capabilities to another party by signing a **"delegation of authority"** to the other party's public key
- With hash signatures, signer can delegate by providing a **subset of its one-time private keys**
  - Delegation scope defined by *index semantics*: what 1, …, *N* mean

- Advantage: Coarse-grained delegation **without a second level of keys**
  - Trust model improvement: Signer can **involve other parties in signing**, while setting (coarse) bounds on their authority
- Applications: Load-balanced / proxy signing with traceable signatures

Coarse-Grained Delegation

Delegation enhancement:

- Delegate **subsets of one-time private keys** to other parties
- Associate indices with **specific meanings or limitations**
  - e.g., second half = “may be delegated”: verifier may treat these differently than first half



A Proposed Adoption Strategy for Hash Signatures

Three Steps toward Adoption

1. Fit into existing framework, but extend with new functionality
   - Hash signature specifications should fit into existing framework for signing, verification, key management, to simplify integration
   - Specifications should also describe new functionality, e.g., short backdating windows, coarse-grained delegation
2. Develop supporting tools, challenge assumptions as needed to leverage new functionality
   - Index-based policies for valid signing times, delegation scope
   - Forensics based on traceable signatures
   - Challenge assumptions: valid signing time = public key validity period; delegation requires fine-grained statement; signatures don’t identify where they were generated
3. Find applications where new functionality matters

A Candidate Application: DNSSEC Signing

- Domain Name System Security Extensions (DNSSEC) add signatures to records for **end-to-end data integrity**
  - Records, signatures returned in response to lookup requests to **name servers**
- Signatures typically **precomputed offline** when records are updated – not in real time
  - Advantage: Reduce risk of private key compromise; name server instances don't need to be trusted to sign
  - Disadvantage: Dynamic range limited to what's been precomputed
- If hash signatures were adopted, signing operations could be **delegated with traceability** to name server instances
- Application question: Would it matter if signatures could be computed in real time in some cases?

Summary

- Long-term advantages hard to sell on their own – need near-term advantages as well
- Hash-based signatures offer significant new functionality
- To sell hash-based signatures, find applications where new functionality matters, focus on these for early adoption
- Operational experience with these applications will facilitate adoption elsewhere in the long term