A Bundesdruckerei Company

Hash-Based Signatures

Stefan-Lukas Gazdag
<Stefan-Lukas_Gazdag@genua.eu>

5th ENISA/FORTH Summer School, 27th of September 2018

Table of contents

Post-Quantum Cryptography

Various flavours:

- Lattice-based cryptography
- Hash-based cryptography
- Code-based cryptography
- Further techniques (e.g. multivariate, isogeny-based, ...)

Post-Quantum Cryptography

Various flavours:

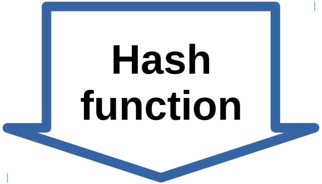
- Lattice-based cryptography
- Hash-based cryptography
- Code-based cryptography
- Further techniques (e.g. multivariate, isogeny-based, ...)

A Bundesdruckerei Company

Basics

Hash functions

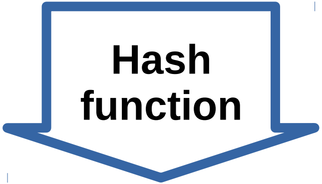
Transfer 1,000,000 USD to bank account 111



7b1df29374728f0aa72d7eaac0d3bdb9dfcb5142111e0e025996dc183ff2caf1
eb529989916758009c87c1244e55944cdded257dcf360caf76c829e93f09811

Hash functions

Transfer 9,000,000 USD to bank account 111

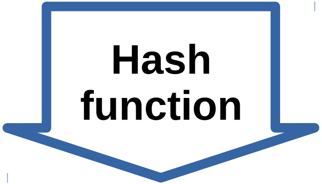


6f2fc9a1ff989bda9ee4e7341c300d29b0e408f5eb977485b32e04bf16b1ca87
b6fb6801e58f1ba8bf5620e1ea12a013b96020b8a47a7e7e6d6c4ccdbc51b7ef

genua A Bundesdruckerei Company

Hash functions

Transfer 1,000,000 USD to bank account 112

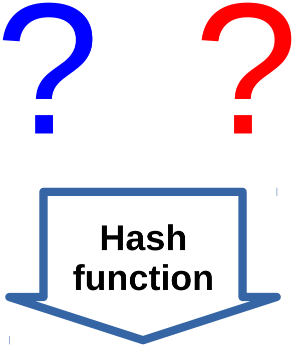


10c9827e0859c7c0abe39deed36386c84652f5a7312ca63fcb5d17f286d25e22
dde90a6f65bd2d4d697ae5c1a57dd42e96260d8f5ff5d7da4211da1868102d6b

Security Properties of Hash Functions

- Pre-image resistance (One-wayness)
- Second pre-image resistance
- Collision resistance

Security Properties: Collision Resistance

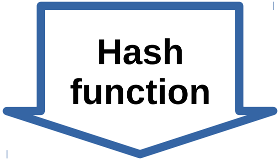


7b1df29374728f0aa72d7eaac0d3bdb9
dfcb5142111e0e025996dc183ff2caf1e
b529989916758009c87c1244e55944
cddded257dcf360caf76c829e93f09811

Security Properties: Second Pre-Image Resistance

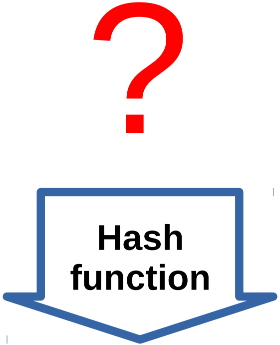
Transfer
1,000,000
USD
to bank
Account
111

?



7b1df29374728f0aa72d7eaac0d3bdb9
dfcb5142111e0e025996dc183ff2caf1e
b529989916758009c87c1244e55944
cddded257dcf360caf76c829e93f09811

Security Properties: Pre-Image Resistance



7b1df29374728f0aa72d7eaac0d3bdb9
dfcb5142111e0e025996dc183ff2caf1e
b529989916758009c87c1244e55944
cddded257dcf360caf76c829e93f09811

Hash-Based Signatures

A suitable solution

Why use hash-based signatures?

- Post-quantum
- Appropriate performance (< 1 ms to a few sec.)
- Data sizes / structures somewhat small enough (ca. 2 to 50 kB for a signature)

A suitable solution

Why use hash-based signatures?

- Post-quantum
- Appropriate performance ($<$ 1 ms to a few sec.)
- Data sizes / structures somewhat small enough (ca. 2 to 50 kB for a signature)
- Limited but suitable life time of the key
- Invented by Ralph C. Merkle and published 1979
- Intense examination and advancement since the 1990s

A suitable solution

Why use hash-based signatures?

- Security of the scheme *only* relies on the security of the hash function

A suitable solution

Why use hash-based signatures?

- Security of the scheme *only* relies on the security of the hash function
- Hash function may be exchanged
  $\Rightarrow$ scheme itself stays secure

A suitable solution

Why use hash-based signatures?

- Security of the scheme *only* relies on the security of the hash function
- Hash function may be exchanged
  $\Rightarrow$ scheme itself stays secure
- We can trust the security already

A suitable solution

Why use hash-based signatures?

- Security of the scheme *only* relies on the security of the hash function
- Hash function may be exchanged
  $\Rightarrow$ scheme itself stays secure
- We can trust the security already
- Second pre-image resistance sufficient for some derivates
  (but still needs further measures like keyed hash function calls)

History repeats itself!

Collision resistance:

- 1992: MD5 published
- 1993 - 2004: Theoretical attacks!
- 2008: Practical attack!

History repeats itself!

Collision resistance:

- 1992: MD5 published
- 1993 - 2004: Theoretical attacks!
- 2008: Practical attack!

- 1993: SHA-1 published
- 2005 - 2015: Theoretical attacks!
- 2017: Practical attack!

History repeats itself!

Collision resistance:

- 1992: MD5 published
- 1993 - 2004: Theoretical attacks!
- 2008: Practical attack!

- 1993: SHA-1 published
- 2005 - 2015: Theoretical attacks!
- 2017: Practical attack!

No attacks by finding a second pre-image for MD5 or SHA-1 by today!

Security

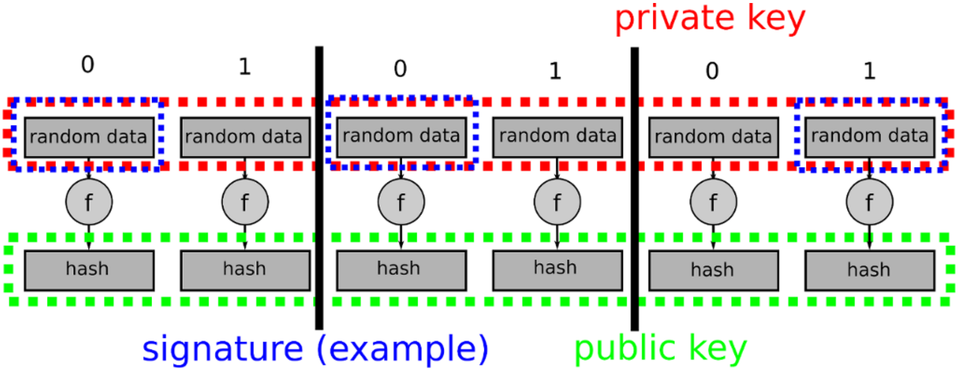
Generic:

Basically a brute-force attack on a list of n keys.

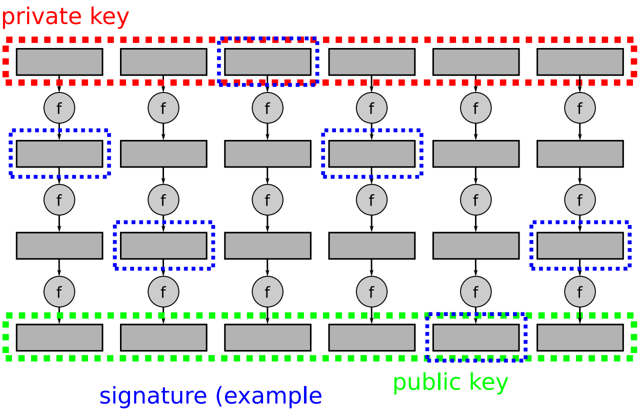
Attack using Grover's algorithm $\Rightarrow \sqrt{n}$

In a quantum setting you got to use SHA-512
if you need the security of SHA-256 in the classical setting.

One-Time Signature Scheme



One-Time Signature Scheme



Verification

What does the receiver get?

- message
- signature
- public / verification key

Verification

What does the receiver get?

- message
- signature
- public / verification key

What does the receiver do?

- Evolve / hash public key according to message
- Check if generated public key is equal to given public key

A Bundesdruckerei Company

Verification

What does the receiver get?

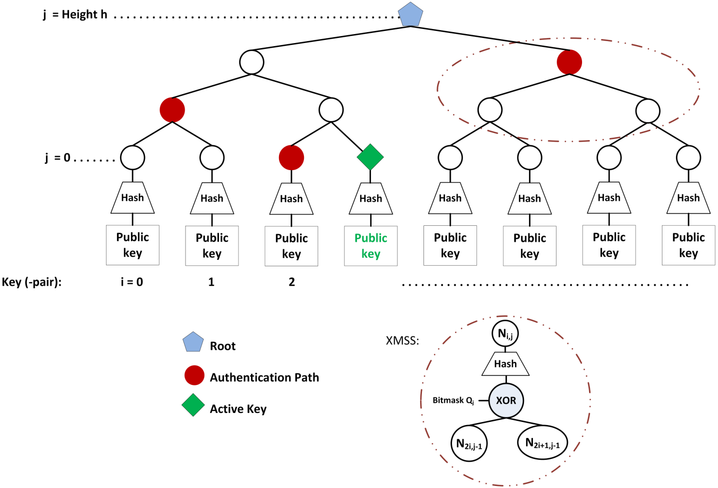
- message
- signature
- public / verification key

What does the receiver do?

- Evolve / hash public key according to message
- Check if generated public key is equal to given public key

How do we exchange the public /verification key?
Or: How do we make sure the sender is authentic?

Merkle Signatures



Verification

What does the receiver get?

- message
- one-time signature
- one-time public / verification key
- authentication path (nodes)

Via a different channel (certificate, ...):

- root of the tree (Merkle public key)

Verification

What does the receiver get?

- message
- one-time signature
- one-time public / verification key
- authentication path (nodes)

Via a different channel (certificate, ...):

- root of the tree (Merkle public key)

What does the receiver do?

- Evolve one-time public key according to message
- One-time public key equal to given one-time public key?
- Calculate leaf and evolve it to root by using authentication path
- Calculated root equal to given root (Merkle public key)?

Verification

What does the receiver get?

- message
- one-time signature
- one-time public / verification key
- authentication path (nodes)

Via a different channel (certificate, ...):

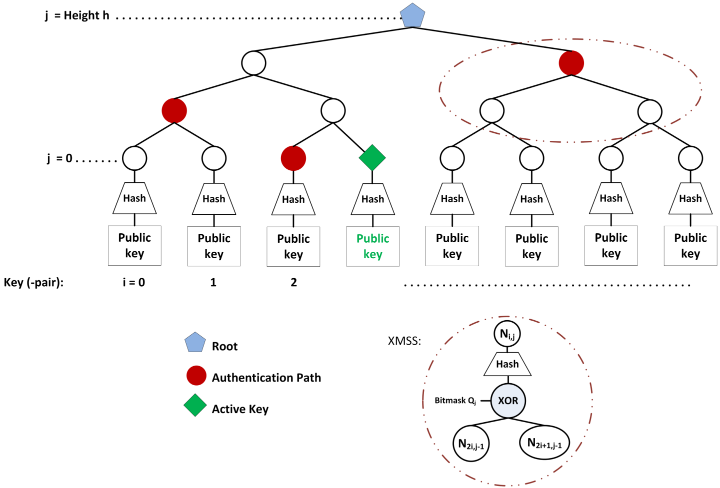
- root of the tree (Merkle public key)

What does the receiver do?

- Evolve one-time public key according to message
- One-time public key equal to given one-time public key?
- Calculate leaf and evolve it to root by using authentication path
- Calculated root equal to given root (Merkle public key)?

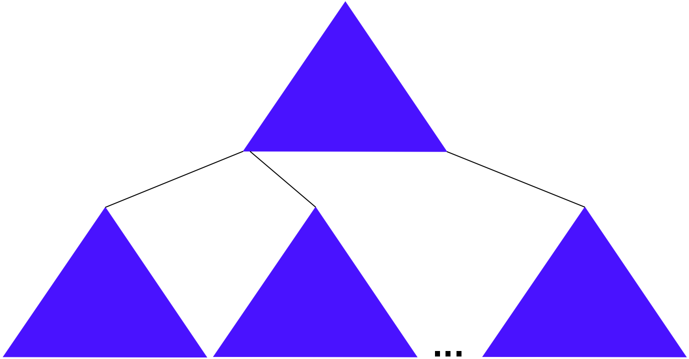
Actually this can be optimized.

Merkle Signatures

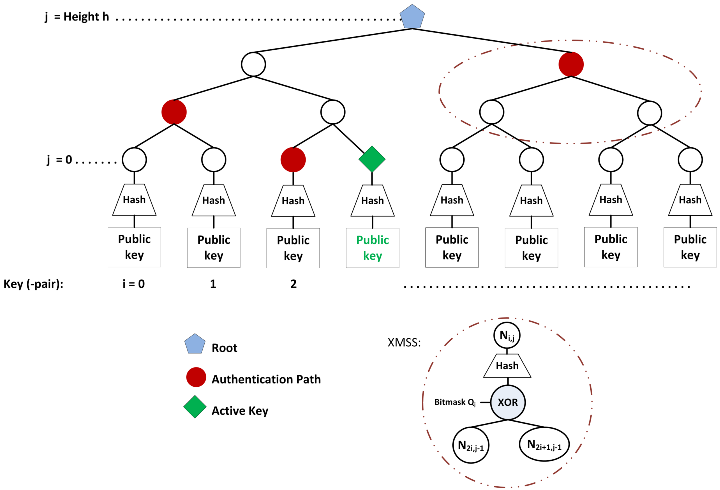


Multiple layers

Multi-tree or hyper-tree



Merkle Signatures



The State

Keep track: which key pairs have not been used yet?

The State

Keep track: which key pairs have not been used yet?

- Integer: next key pair
- If there's a state anyway let's
  - generate one-time key pairs with PRNG
  - only store part of the tree

The State

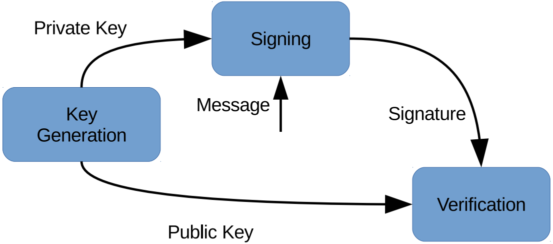
Keep track: which key pairs have not been used yet?

- Integer: next key pair
- If there's a state anyway let's
  - generate one-time key pairs with PRNG
  - only store part of the tree

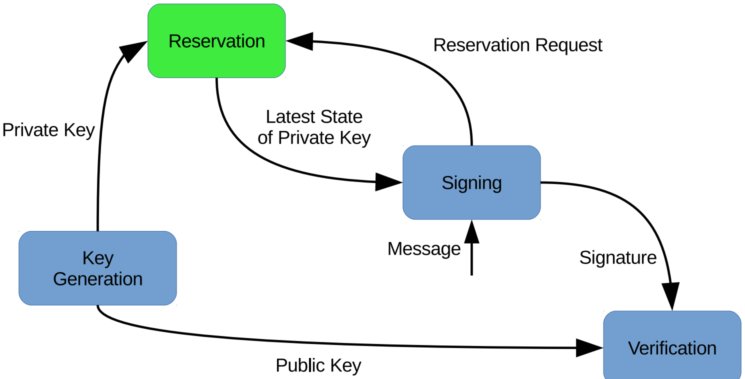
Side effects:

- Secret key becomes critical resource!
- Copies of the key may leak old state!

Classical signatures



Reservation Approach



A Bundesdruckerei Company

State Mangement

State Management for Hash-Based Signatures

David McGrew[1], Panos Kampanakis[1], Scott Fluhrer[1],
Stefan-Lukas Gazdag[2], Denis Butin[3], and Johannes Buchmann[3]

[1] Cisco Systems, USA
{mcgrew,pkampana,sfluhrer}@cisco.com
[2] genua GmbH, Germany
stefan-lukas_gazdag@genua.eu
[3] TU Darmstadt, Germany
{dbutin,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. The unavoidable transition to post-quantum cryptography requires dependable quantum-safe digital signature schemes. Hash-based signatures are well-understood and promising candidates, and the object of current standardization efforts. In the scope of this standardization process, the most commonly raised concern is statefulness, due to the use of one-time signature schemes. While the theory of hash-based signatures is mature, a discussion of the system security issues arising from the concrete management of their state has been lacking. In this paper, we analyze state management in N-time hash-based signature schemes, considering both security and performance, and categorize the security issues that can occur due to state synchronization failures. We describe a state reservation approach that loosens the coupling between volatile and nonvolatile storage, and show that it can be naturally realized in a hierarchical signature scheme. To protect against unintentional copying of the private key state, we consider a hybrid stateless/stateful scheme, which provides a graceful security degradation in the face of unintentional copying, at the cost of increased signature size. Compared to a completely stateless scheme, the hybrid approach realizes the essential benefits, with smaller signatures and faster signing.

McGrew et al., *State Management for Hash-Based Signatures*, SSR 2016, Springer LNCS 10074

A Bundesdruckerei Company

Going Stateless

May we omit the state?

Going Stateless

May we omit the state?

$\Rightarrow$ Yes, if trusting probabilites.

Going Stateless

May we omit the state?

$\Rightarrow$ Yes, if trusting probabilites.

Basic idea:
Use a tree so huge you can randomly choose a one-time key pair.

Going Stateless

May we omit the state?

$\Rightarrow$ Yes, if trusting probabilites.

Basic idea:
Use a tree so huge you can randomly choose a one-time key pair.

Use a big hyper-tree and few-time key pairs!

Going Stateless

May we omit the state?

$\Rightarrow$ Yes, if trusting probabilites.

Basic idea:
Use a tree so huge you can randomly choose a one-time key pair.

Use a big hyper-tree and few-time key pairs!

Bernstein et al., *SPHINCS: practical stateless hash-based signatures*, EUROCRYPT 2015, Springer LNCS 9056

Standardization

Schemes in standardization

IETF/IRTF:

- XMSS and XMSS^{MT}

  $\Rightarrow$ Published as RFC 8391

- LMS and HSS

  $\Rightarrow$ Soon to be published as RFC

NIST:

- SPHINCS^{+}

  $\Rightarrow$ Candidate for NIST standardization

- Gravity-SPHINCS

  $\Rightarrow$ Candidate for NIST standardization

IETF/IRTF RFC

Internet Research Task Force (IRTF)
Request for Comments: 8391
Category: Informational
ISSN: 2070-1721

A. Huelsing
TU Eindhoven
D. Butin
TU Darmstadt
S. Gazdag
genua GmbH
J. Rijneveld
Radboud University
A. Mohaisen
University of Central Florida
May 2018

XMSS: eXtended Merkle Signature Scheme

Abstract

This note describes the eXtended Merkle Signature Scheme (XMSS), a hash-based digital signature system that is based on existing descriptions in scientific literature. This note specifies Winternitz One-Time Signature Plus (WOTS+), a one-time signature scheme; XMSS, a single-tree scheme; and XMSS^MT, a multi-tree variant of XMSS. Both XMSS and XMSS^MT use WOTS+ as a main building block. XMSS provides cryptographic digital signatures without relying on the conjectured hardness of mathematical problems. Instead, it is proven that it only relies on the properties of cryptographic hash functions. XMSS provides strong security guarantees and is even secure when the collision resistance of the underlying hash function is broken. It is suitable for compact implementations, is relatively simple to implement, and naturally resists side-channel attacks. Unlike most other signature systems, hash-based signatures can so far withstand known attacks using quantum computers.

A Bundesdruckerei Company

IETF/IRTF Internet-Draft

Crypto Forum Research Group — D. McGrew
Internet-Draft — M. Curcio
Intended status: Informational — S. Fluhrer
Expires: March 10, 2019 — Cisco Systems
September 6, 2018

Hash-Based Signatures
draft-mcgrew-hash-sigs-13

Abstract

This note describes a digital signature system based on cryptographic hash functions, following the seminal work in this area of Lamport, Diffie, Winternitz, and Merkle, as adapted by Leighton and Micali in 1995. It specifies a one-time signature scheme and a general signature scheme. These systems provide asymmetric authentication without using large integer mathematics and can achieve a high security level. They are suitable for compact implementations, are relatively simple to implement, and naturally resist side-channel attacks. Unlike most other signature systems, hash-based signatures would still be secure even if it proves feasible for an attacker to build a quantum computer.

This document is a product of the Crypto Forum Research Group (CFRG) in the IRTF.

NIST Process - HBS

Q: What are NIST's plans regarding stateful hash-based signatures?

A: NIST plans to coordinate with other standards organizations, such as the IETF, to develop standards for stateful hash-based signatures. As stateful hash-based signatures do not meet the API requested for signatures, this standardization effort will be a separate process from the one outlined in the call for proposals. It is expected that NIST will only approve a stateful hash-based signature standard for use in a limited range of signature applications, such as code signing, where most implementations will be able to securely deal with the requirement to keep state.

https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/faqs

NIST Process - HBS

Gravity-SPHINCS	Zip File (8MB) KAT Files (36MB) IP Statements Website	Jean-Phillippe Aumasson Guillaume Endignoux	Submit Comment View Comments

SPHINCS+	Zip File (2MB) KAT Files (61MB) IP Statements Website	Andreas Hulsing Daniel J. Bernstein Christoph Dobraunig Maria Eichlseder Scott Fluhrer Stefan-Lukas Gazdag Panos Kampanakis Stefan Kolbl Tanja Lange Martin M Lauridsen Florian Mendel Ruben Niederhagen Christian Rechberger Joost Rijneveld Peter Schwabe	Submit Comment View Comments

https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions

BSI

Bundesamt für Sicherheit in der Informationstechnik

BSI – Technische Richtlinie

Bezeichnung:	**Kryptographische Verfahren: Empfehlungen und Schlüssellängen**
Kürzel:	BSI TR-02102-1
Version:	2017-01
Stand:	8. Februar 2017

5.4.4. Merklesignaturen

Im Gegensatz zu den bisher beschriebenen Signaturverfahren beruht die Sicherheit des in [30] beschriebenen Algorithmus nur auf der kryptographischen Stärke einer Hashfunktion und einer pseudozufälligen Funktionenfamilie. Insbesondere werden keine Annahmen zur Abwesenheit effizienter Lösungsalgorithmen für Probleme aus der algorithmischen Zahlentheorie wie das RSA-Problem oder die Berechnung diskreter Logarithmen benötigt. Es wird deshalb allgemein angenommen, dass Merklesignaturen im Gegensatz zu allen anderen in dieser Technischen Richtlinie empfohlenen Signaturverfahren auch gegen Angriffe unter Verwendung von Quantencomputern sicher bleiben würden.[2]

Als Hashfunktionen sind alle in Tabelle 4.1 empfohlenen Hashverfahren geeignet. Die benötigte pseudozufällige Funktionenfamilie kann durch die HMAC-Konstruktion aus der verwendeten Hashfunktion konstruiert werden.

Für eine genaue Beschreibung des Verfahrens siehe [30].

Die generell geringen komplexitätstheoretischen Annahmen, die der Sicherheit von Merkle-Signaturen zugrundeliegen, lassen Merkle-Signaturen als eine gute Methode für die Erstellung langfristig sicherer Signaturen erscheinen. Dies gilt auch unter der Annahme, dass Angriffe durch Quantencomputer über den Zeitraum hinweg, in dem die Signatur gültig bleiben soll, keine Anwendung finden.

Anders als in den anderen in der vorliegenden Technischen Richtlinie beschriebenen Signaturverfahren kann bei Verwendung von Merkle-Signaturen mit einem gegebenen öffentlichen Schlüssel allerdings jeweils nur eine endliche Anzahl von Nachrichten authentifiziert werden. Außerdem ist die Rechenzeit zur Erzeugung des öffentlichen Schlüssels proportional zu dieser Anzahl zu authentisierender Nachrichten und damit vergleichsweise lang, wenn eine große Anzahl von Nachrichten ohne zwischenzeitliche Erzeugung und authentisierte Verteilung eines neuen öffentlichen Schlüssels signiert werden soll. Ergebnisse praktischer Experimente zur Effizienz aller Teilschritte (Schlüsselgenerierung, Signaturerzeugung, Signaturverifikation) des in [30] beschriebenen Verfahrens und zu den auftretenden Schlüssellängen und Signaturgrößen finden sich in Abschnitt 6 von [30].

Use Cases

A Bundesdruckerei Company

Welcome to the crypto apocalypse

How do you verify updates in the quantum era?

Welcome to the crypto apocalypse

How do you verify updates in the quantum era?

Manufacturer gave you a public key e.g.
by handing you a sealed product.

genua A Bundesdruckerei Company

Welcome to the crypto apocalypse

How do you verify updates in the quantum era?

Manufacturer gave you a public key e.g.
by handing you a sealed product.

Practical quantum computers available?
You can't trust this key anymore!

A Bundesdruckerei Company

Welcome to the crypto apocalypse

How do you verify updates in the quantum era?

Manufacturer gave you a public key e.g.
by handing you a sealed product.

Practical quantum computers available?
You can't trust this key anymore!

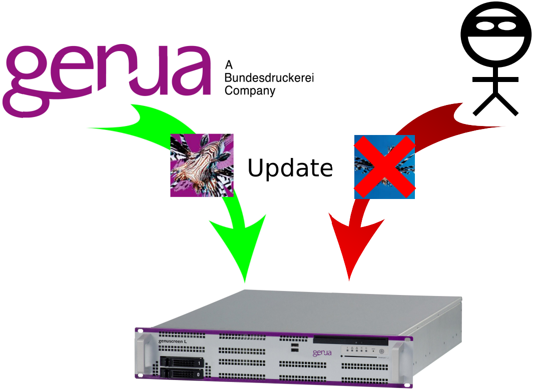
Want to do a recall? In IoT scale?
A mounted messenger handing you a new key?

Update Signatures

Fairly easy to handle:

- Dedicated key server
- Restricted environment
- Manageable number of signatures
- Acceptable timing / size restrictions (more or less)
- *Hybrid* signature release

Update Signatures



First products provided with a post-quantum update signature available!

A Bundesdruckerei Company

Use cases for HBS

Update signatures (code signing) are the perfect use case for HBSs.

What else?

- SSH somewhat ok (XMSS available in OpenSSH)
- PKI somewhat ok
- S/MIME / e-mail somewhat ok
- TLS not that much (though some people would object)

Most importantly (and critical): Where are the keys handled and stored?
⇒ Best solutions are smartcards or hardware security modules.

Conclusion

- We can use hash-based signatures already!
- Not suitable for every use case,
  but convenient for several important ones.
- Different settings demand different keys,
  but more and more experience is gained.

Questions?

Stefan-Lukas_Gazdag@genua.eu

www.square-up.org