accu 2021
VIRTUAL EVENT

Bloomberg Engineering

undo

mosaic
CONSULTANTS TO FINANCIAL SERVICES

How to Build Digital Signatures From Hash Functions

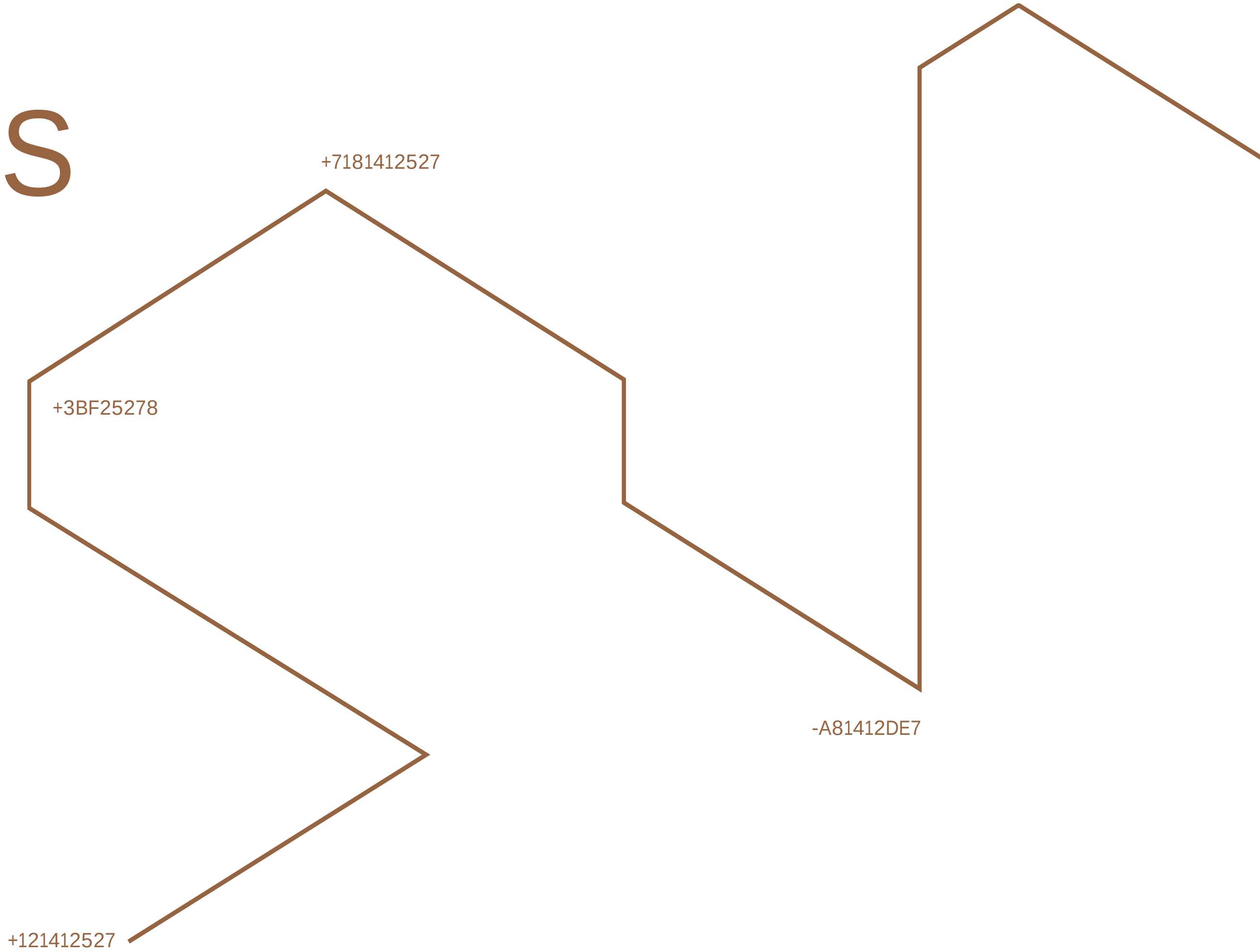
Ahto Truu

guardtime

HOW TO BUILD DIGITAL SIGNATURES FROM HASH FUNCTIONS

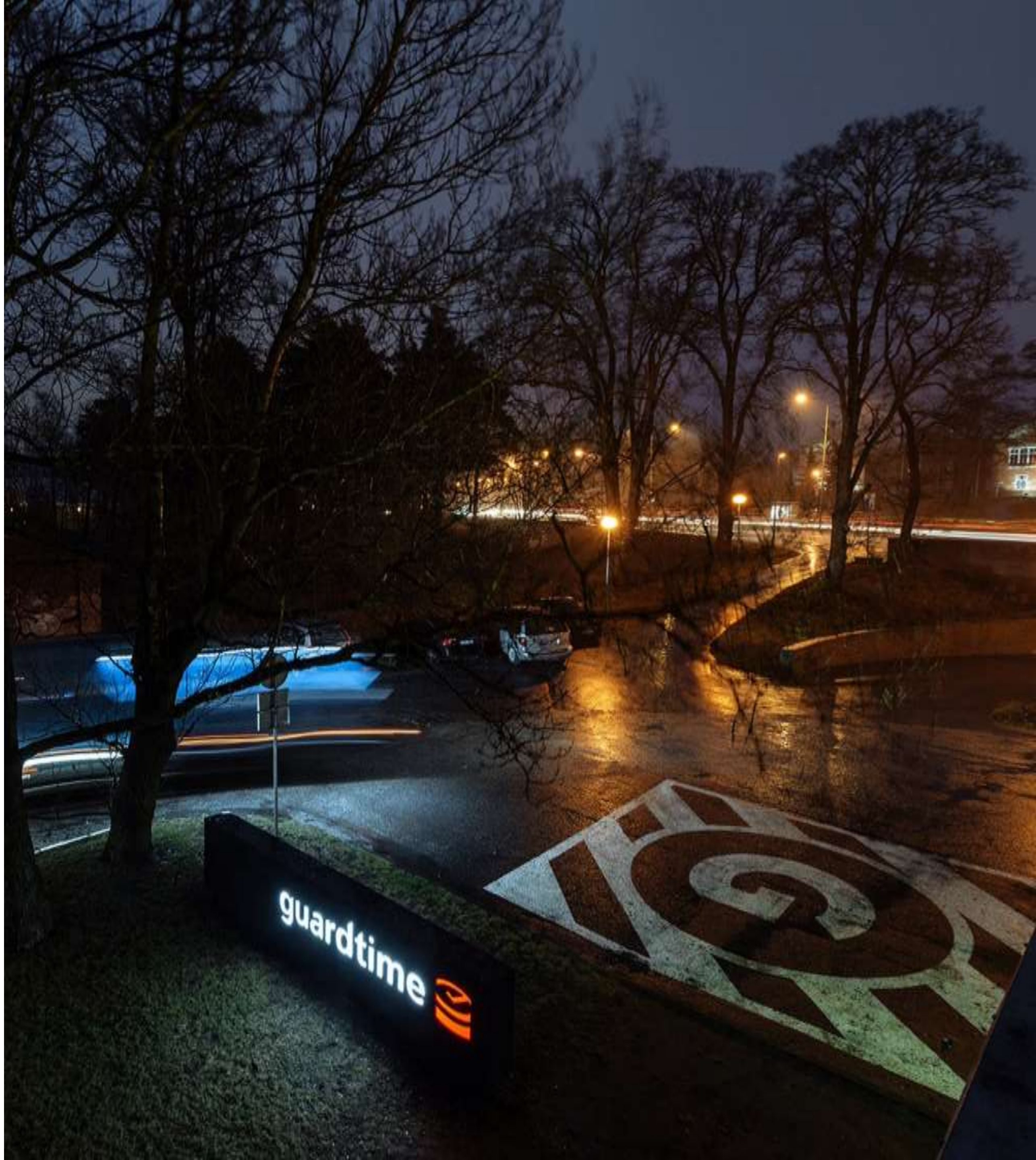
AHTO TRUU
SOFTWARE ARCHITECT, GUARDTIME

ACCU CONFERENCE, 12-MAR-2021



ABOUT GUARDTIME

- Systems engineering company focusing on data security solutions
- Founded in 2007 in Tallinn, Estonia
- Global HQ in Lausanne, Switzerland
- Offices in US, EU and China
- 150 employees
- 80% engineers and researchers
- https://guardtime.com/

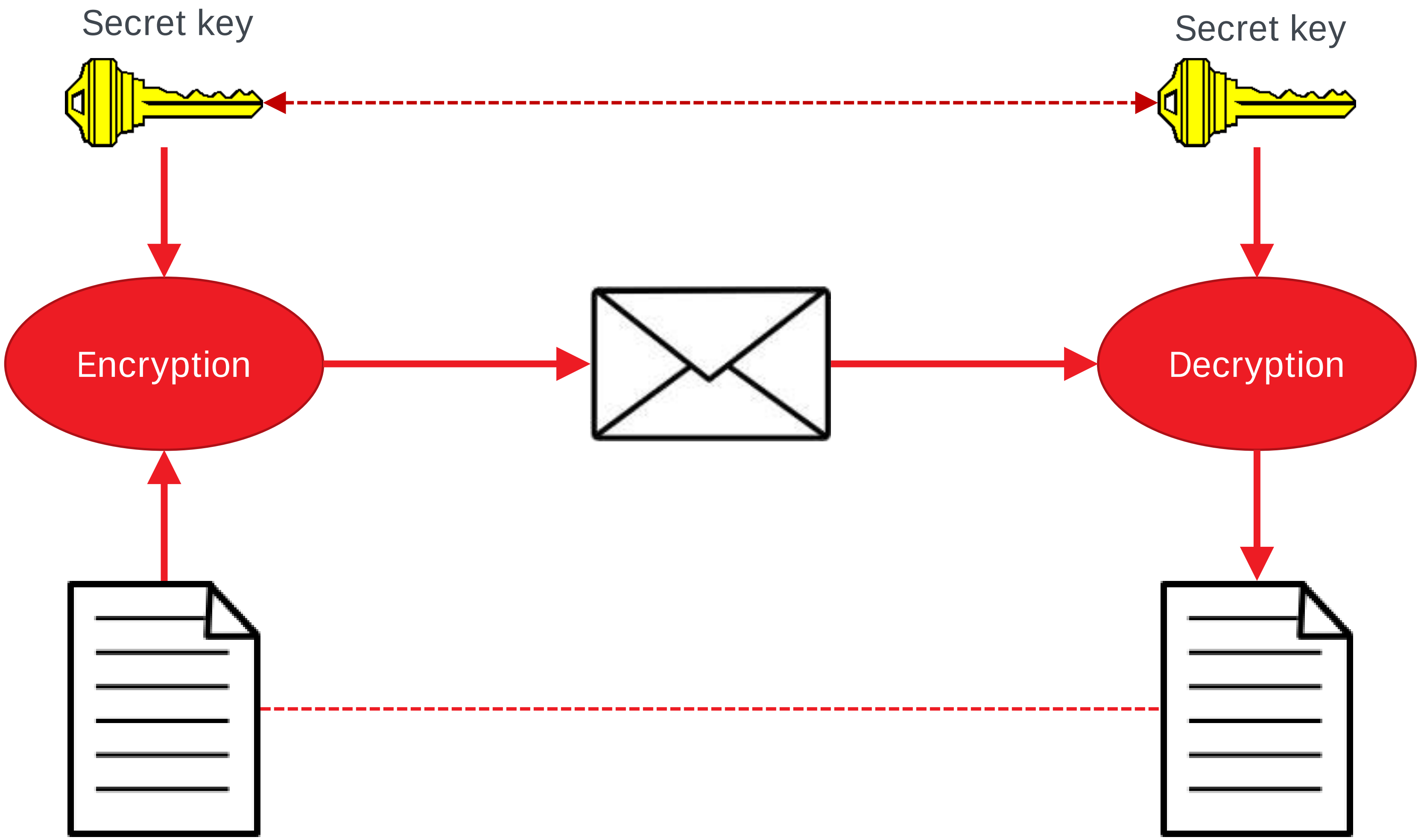


AGENDA

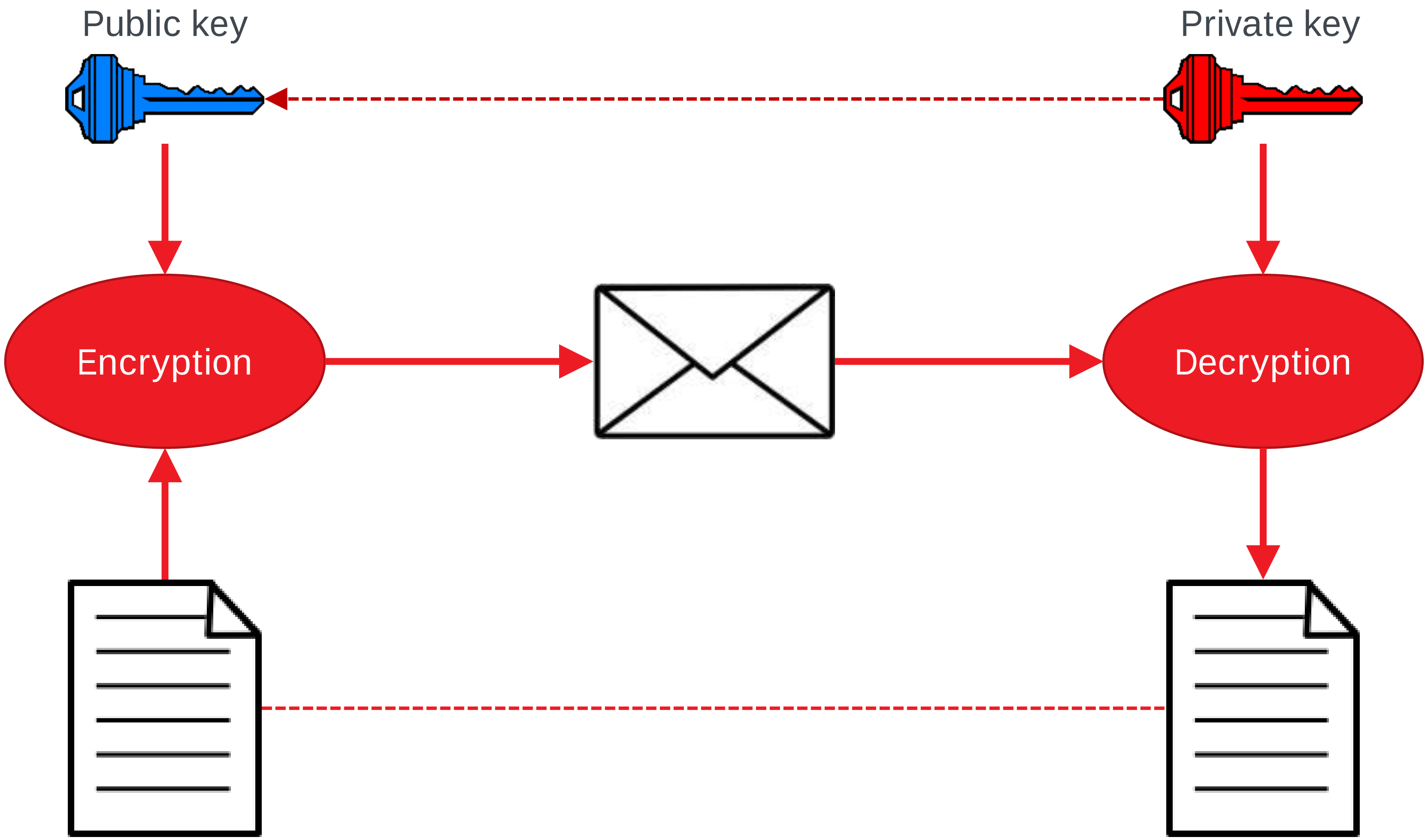
+ Introduction
+ Digital signatures
+ Hash functions
+ Hash signatures
+ Time-stamping
+ BLT signatures

1/
INTRODUCTION

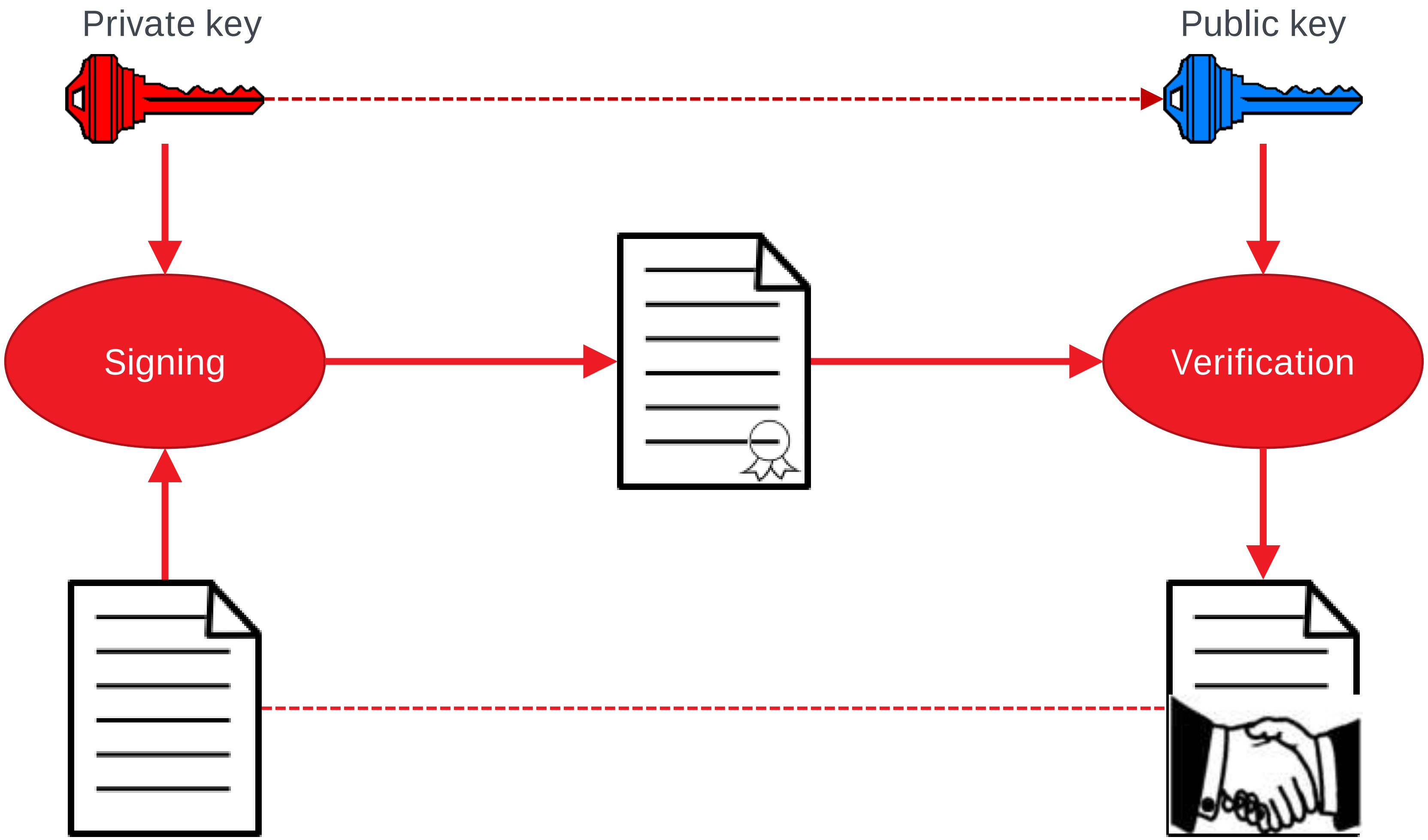
SYMMETRIC ENCRYPTION



ASYMMETRIC ENCRYPTION



DIGITAL SIGNATURES



guardtime

2/
DIGITAL SIGNATURES

DIGITAL SIGNATURES: WHAT ARE THEY

Aim to be the electronic equivalent of hand-written signatures

- Intent: signer's endorsement of the content

- Integrity: authenticity of content
- Identity: authenticity of origin
- Time: authenticity of signing time

- Non-repudiation: signer can't deny a signature afterwards

DIGITAL SIGNATURES: USE-CASES

Relying party (recipient of a signed document) can:

- Verify the authenticity of the document for themselves
- Prove the authenticity of the document to third parties

Main use case types:

- Document signing
- Access control

DIGITAL SIGNATURES: MATHEMATICAL MODEL

guardtime

+ Each signer has two related keys:
  - Private key for creating signatures
  - Public key for verifying signatures

+ A signature scheme consists of three algorithms:
  - Key generation: creates a pair of related keys
  - Signing: gets a document and a private key and creates a signature
  - Verifying: gets a document, a signature, and a public key; checks whether the signature was created with the private key corresponding to the public key

DIGITAL SIGNATURES: SECURITY MODEL

+ For a signature scheme to be secure, it must be infeasible for an attacker to:
  - Change the document without making the verification fail
  - Derive the private signing key from the public verification key
  - Create a signature without access to the private key

+ Also need to make sure unauthorized parties can't access the private key:
  - Best practice to generate the key pair in a secure hardware module
  - Only the public key is exported, the private key never leaves the module
  - For signing, data is sent to the module and signature exported

DIGITAL SIGNATURES: SIGNER IDENTITY

guardtime

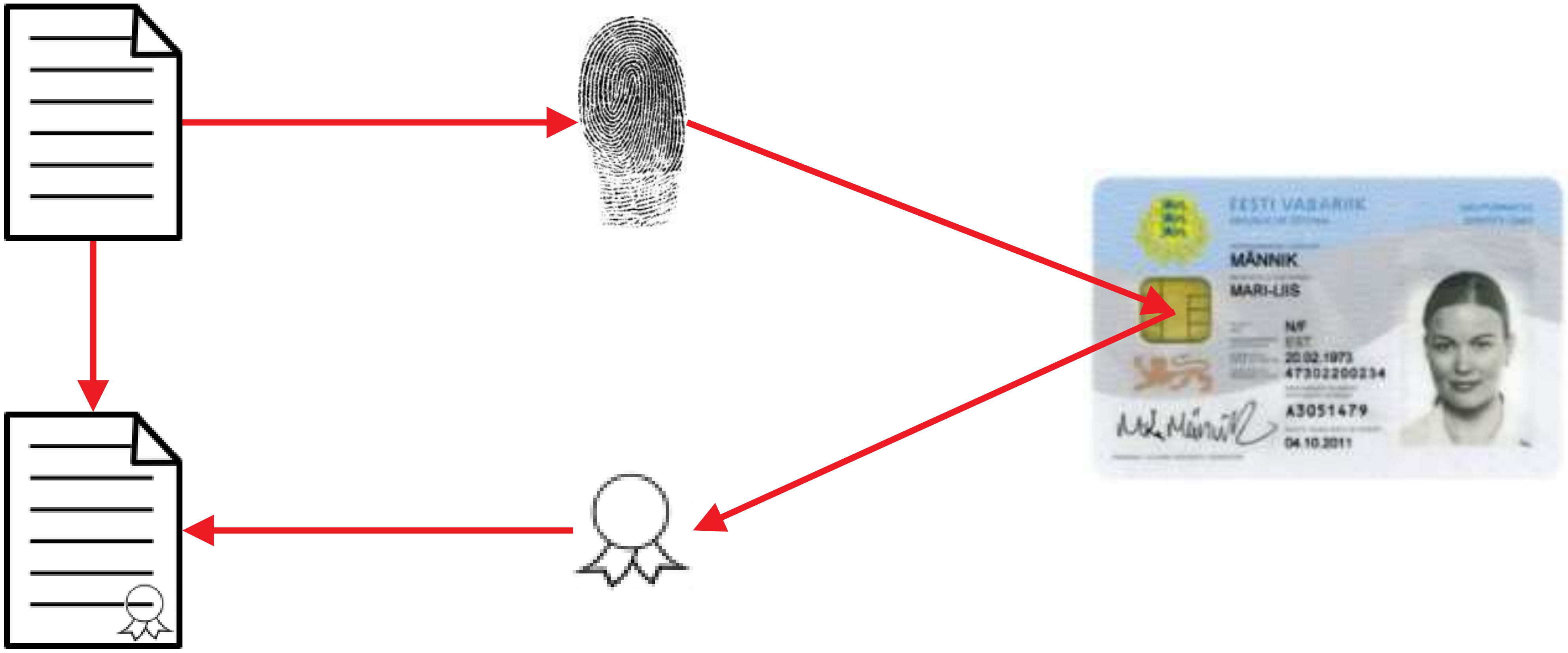
+ A public key is just a piece of data: large random-looking number
+ To authenticate the origin, public keys must be bound to the identities of their holders:
  - The key holder hands the public key directly to the relying party: Mostly used in access control systems
  - Identity of key holder witnessed by someone the relying party knows: Used in the PGP web of trust system, for example
  - Identity of key holder witnessed by a designated authority: Used in the PKI model, where certificates are statements binding public keys to the identities of their holders, signed by dedicated certificate authorities

DIGITAL SIGNATURES: SIGNING TIME

+ Often, digital signature systems must also prove signing time:
- For legal reasons:
  - In most cases a signature is only valid if created in a specific time frame
    - For example, a board member's authority to sign on behalf of the company
- For technical reasons:
  - When an unauthorized party gets the private key, the key must be revoked
  - But this should not be a way for the key holder to disown all previous signatures
  - Need to be able to distinguish between signatures created before and after the key was revoked
- Usually done with the help of time-stamping services

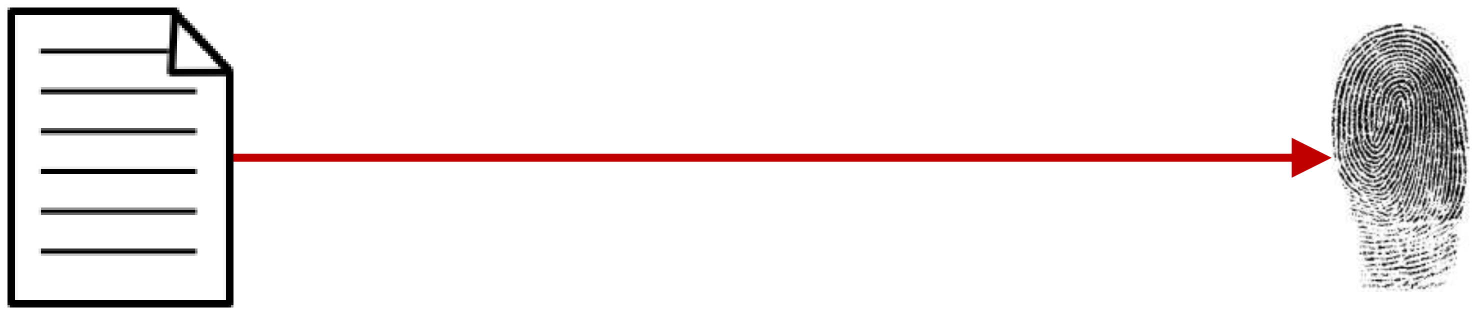
3/ HASH FUNCTIONS

DIGITAL SIGNATURES IN PRACTICE



guardtime

HASH FUNCTIONS



PROPERTIES OF HASH FUNCTIONS

+ Efficiently computable
  - Given x, easy to compute y = f(x)
+ Pre-image resistant
  - Given y, infeasible to find x such that f(x) = y
+ Second pre-image resistant
  - Given x, infeasible to find x’ ≠ x such that f(x’) = f(x)
+ Collision resistant
  - Infeasible to find $x_1 \neq x_2$ such that $f(x_1) = f(x_2)$

guardtime

SECOND PRE-IMAGE RESISTANCE

Bob

- Creates the contract X
- Signs it via h(X)
- Gives it to Alice

Alice

- Modifies X to X’,
  with h(X’) = h(X)
- Claims Bob signed X’

Forgery after signing

X = “I owe Alice $10”

X’ = “I owe Alice $1000”

h(X) = h(X’)

COLLISION RESISTANCE

Alice

- Creates X_1 and X_2 with $h(X_1) = h(X_2)$

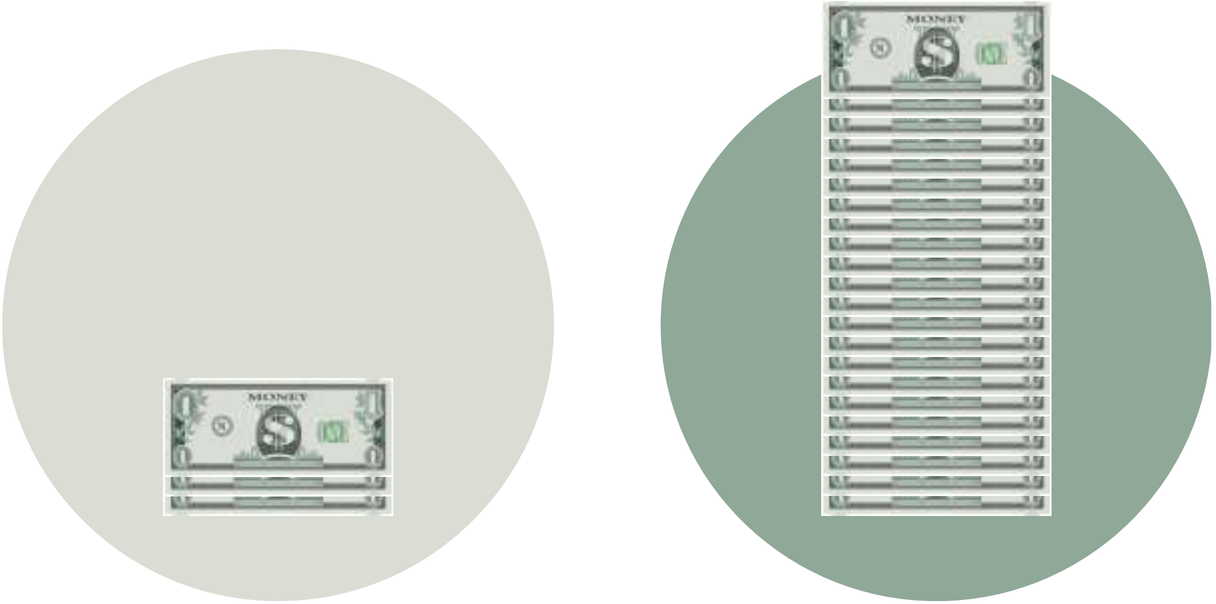
Bob

- Signs X_1 via $h(X_1)$

Alice

- Claims Bob signed X_2

Forgery before signing



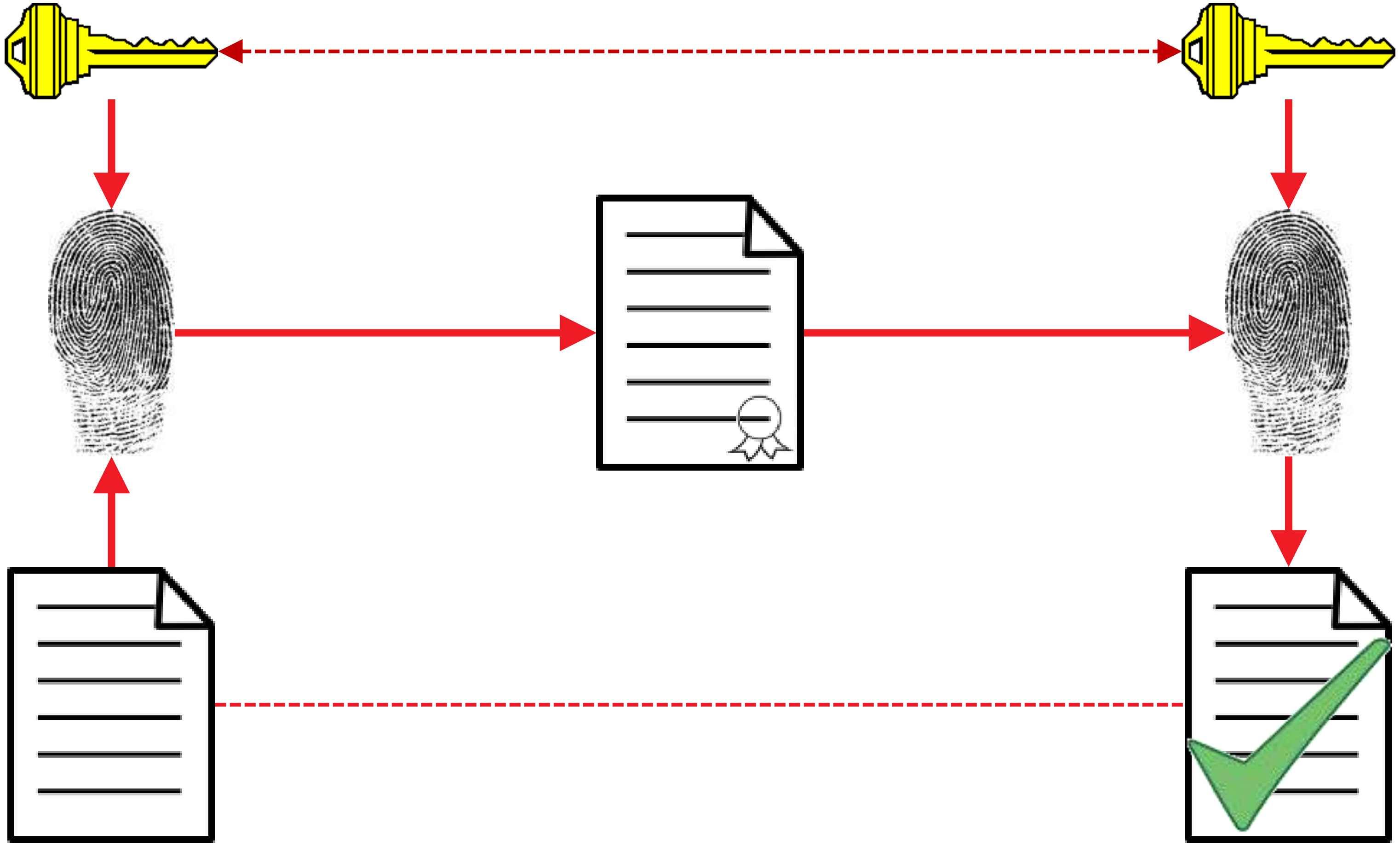
X_1 = “Bob owes me $10” X_2 = “Bob owes me $1000”

$h(X_1) = h(X_2)$

guardtime

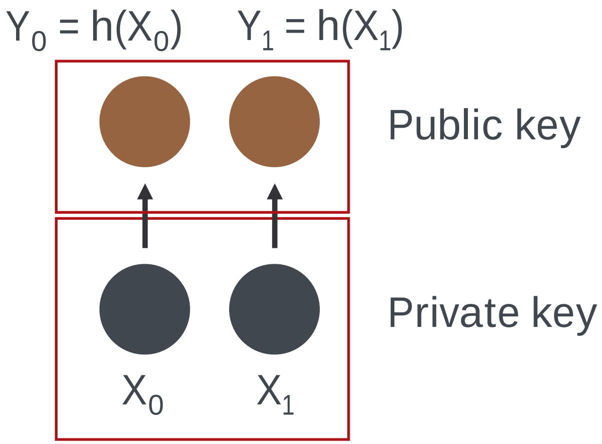
4/
HASH SIGNATURES

MESSAGE AUTHENTICATION CODES



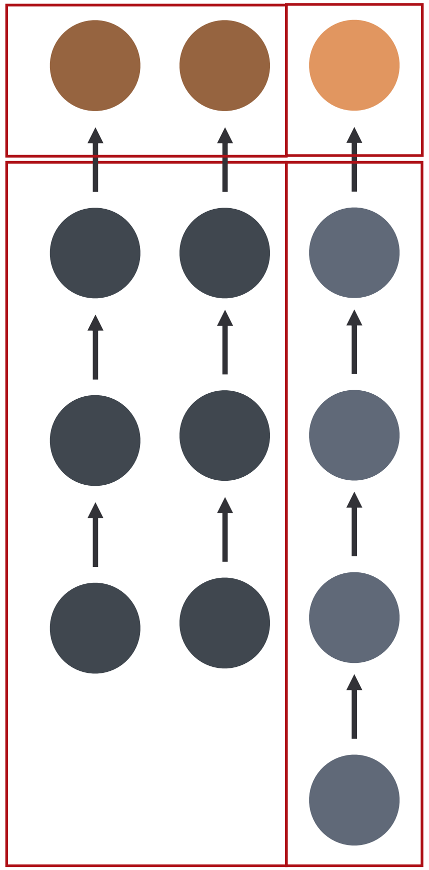
LAMPORT SIGNATURES

+ 1-bit case
  - Private key (X_0, X_1)
  - Public key (Y_0, Y_1)
  - Signature on 0: X_0 (destroy X_1)
  - Signature on 1: X_1 (destroy X_0)
+ Longer inputs
  - Sign each bit separately



WINTERNITZ SIGNATURES

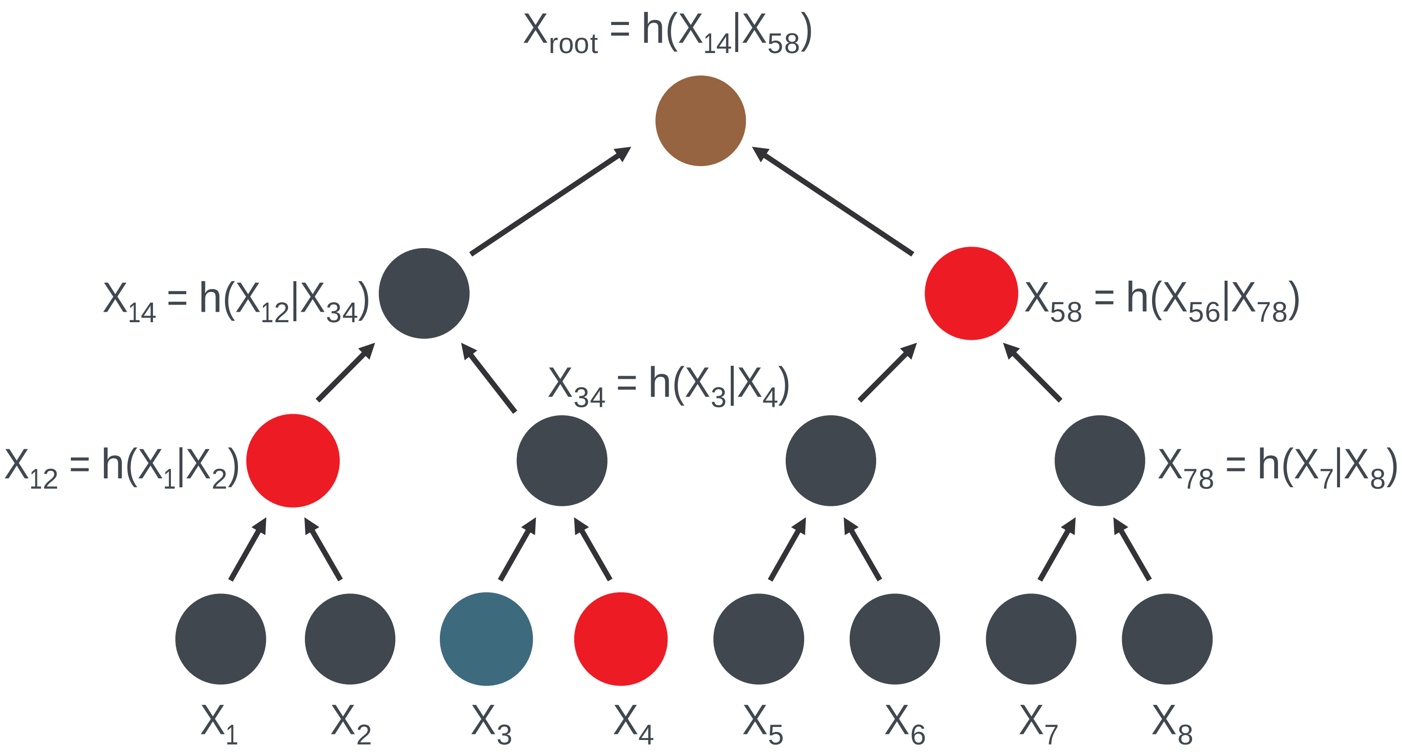
+ W-bit groups
  - 2^W-step hash chains
    $X_i = h(X_{i-1})$
  - Signature on value k: X_k
+ Checksum
  - Sign the total number of steps to public key components



MERKLE TREES

+ A hash tree, or a Merkle tree, aggregates many inputs into a single hash value

+ Afterwards, a compact proof of participation can be extracted for each input



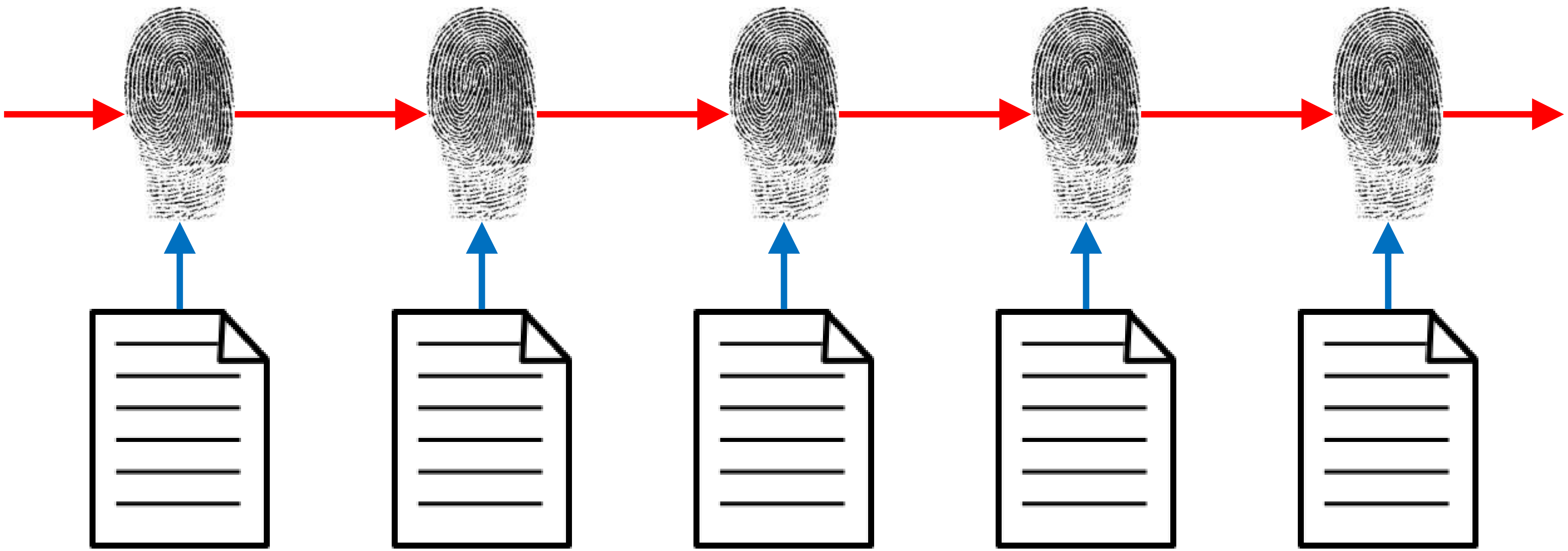
guardtime

5/ TIME-STAMPING

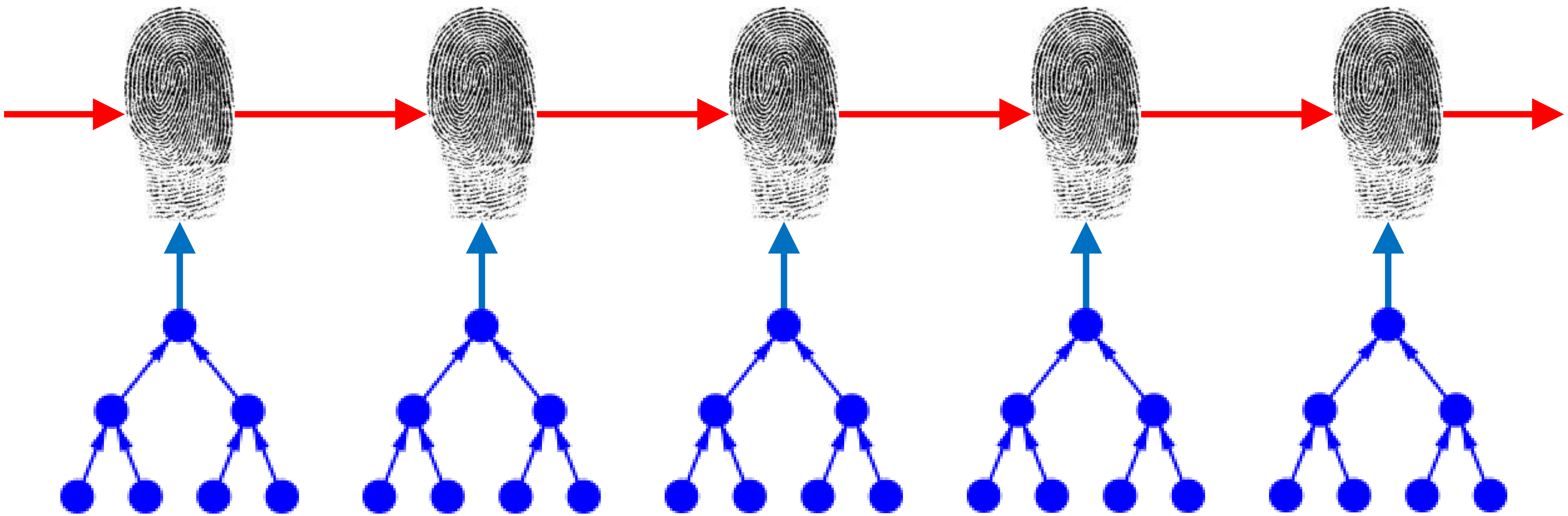
HASH-THEN-PUBLISH TIME-STAMPING

+ To prove the existence of some information: publish it
+ If the information is confidential: publish a hash instead
  - Can later reveal the information and show it matches the hash
+ Galileo, Hooke

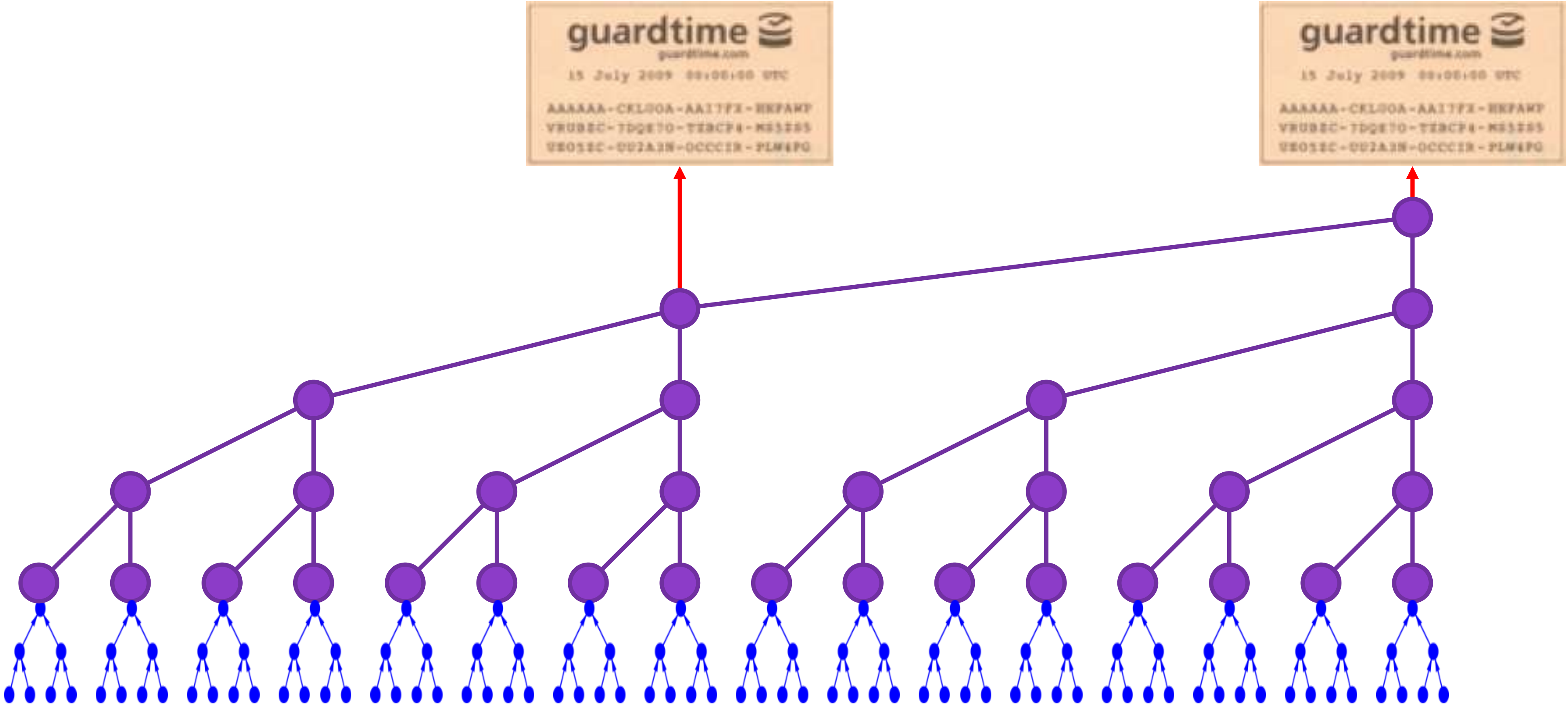
PUBLISHING TO HASH-LINKED LEDGER



HASH-TREE AGGREGATION OF INPUTS



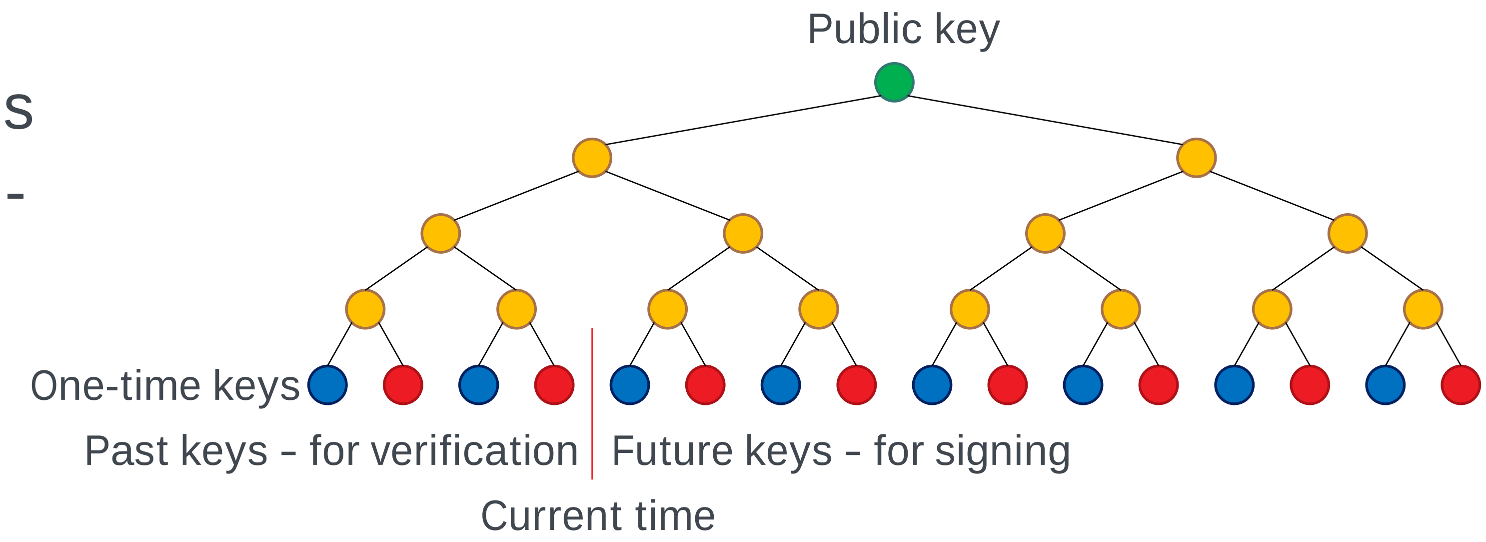
HASH-TREE AGGREGATION OF LEDGER



6/
BLT SIGNATURES

BLT-TB: TIME-BOUND KEYS

+ One-time keys for message authentication
+ Message authentication is symmetric and lacks non-repudiation
+ Use time to break the symmetry

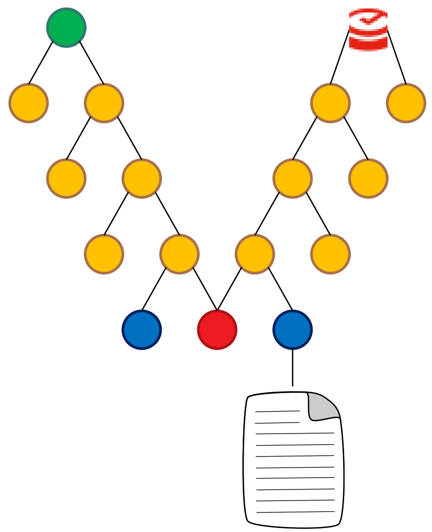


guardtime

BLT-TB: SIGNING

To sign a document at a given time:

- Authenticate the document with the corresponding one-time key
- Time-stamp the authenticator to prove the signing time
- Include proof of the key-time binding in the signature

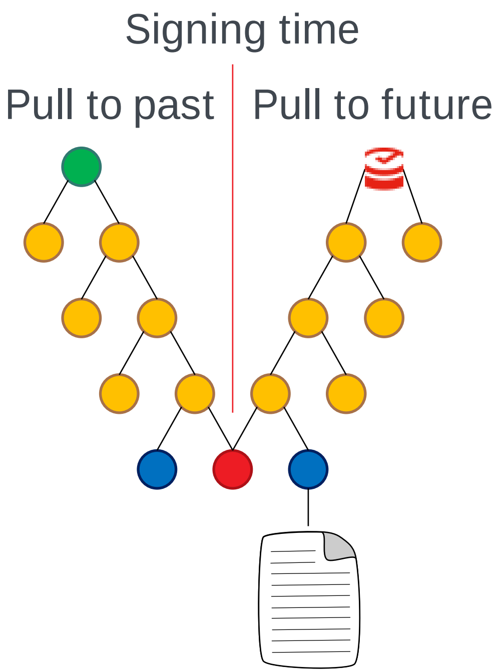


BLT-TB PROPERTIES

- Keys pre-generated for each possible signing time
- Suitable for full-size computers that have
  - Reasonable computing power
  - Reliable clocks
  - Direct network access
- Suitable for applications that need to sign often
  - Ideal for server applications used by many clients

NEW CONCEPT: FORWARD-RESISTANT TAGS

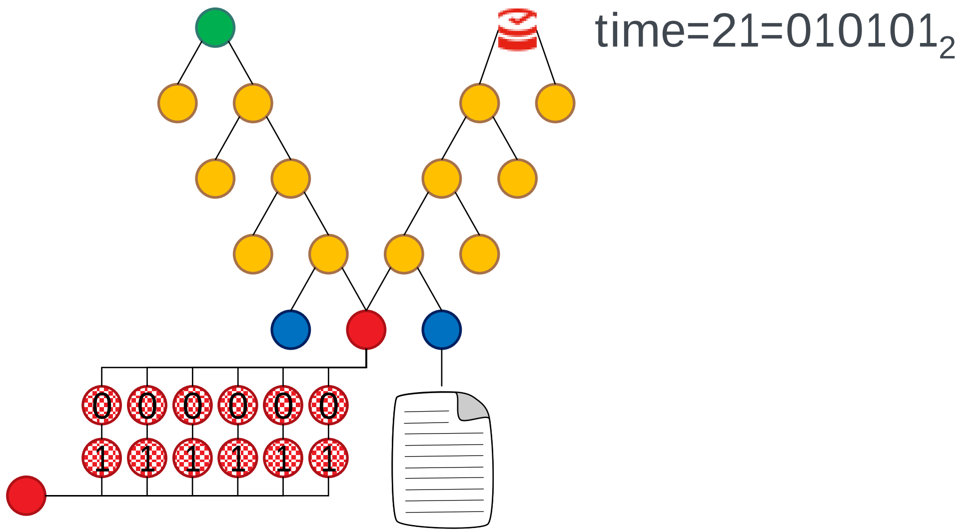
- Pre-binding keys to time slots is wasteful
- Time-stamp already prevents moving the signing event to past
- So, the key binding only needs to prevent moving to future
- So, we can relax the requirement on the key binding



BLT-OT: ONE-TIME KEYS

Inspired by Lamport's signatures

- Generate a multi-component private key
- Bind it to time after time-stamping
- The time value is shorter than a hash value, so less components are needed
- Extra optimization: generate all the key components from a single seed
- Can pre-generate several such keys and use them in sequence



REFERENCES

+ BLT-TB scheme in more detail
  https://eprint.iacr.org/2019/671
+ BLT-OT scheme in more detail
  https://eprint.iacr.org/2019/673

THANK YOU

QUESTIONS?

AHTO.TRUU@GUARDTIME.COM
@AHTOTRUU