Crypto Concepts

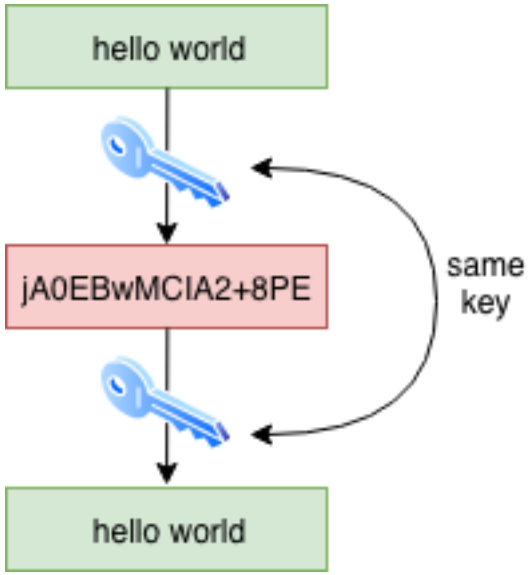
An Introduction

Topics

- Symmetric Encryption
- Asymmetric Encryption
- Digital Signatures
- Hashes
- Certificates

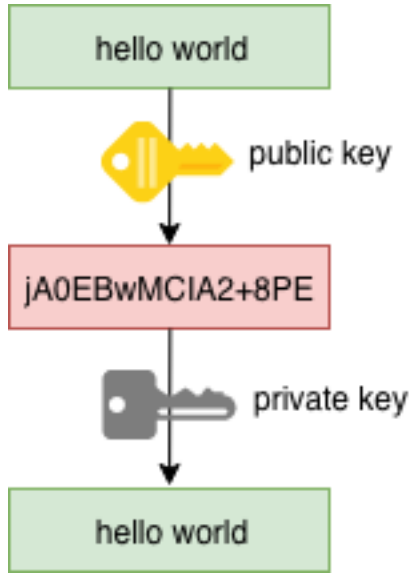
Symmetric Encryption

- A single encryption key encrypts and decrypts a message.
- Similar in concept to a physical key that can lock and unlock a door.
- Pros
  - Conceptually simple.
  - Computationally efficient.
- Cons
  - When transmitting encrypted information, there is the problem of transmitting the key.
  - This is usually not a problem between two individuals who already know each other.
  - It does not scale as the number of parties increases.



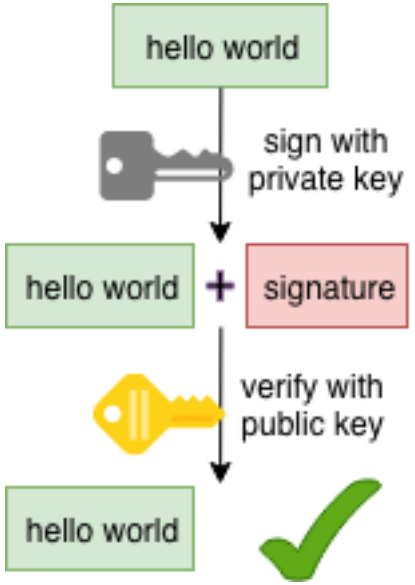
Asymmetric Encryption

- Uses a key-pair.
- One key encrypts, the other decrypts.
  - may start with either key
  - but only the other key will decrypt
  - conventionally assigned as public key and private key.
  - anyone with a public key can encrypt; only the private key holder can decrypt.
- Pros
  - Public key not compromised by disclosure.
- Cons
  - Computationally inefficient.



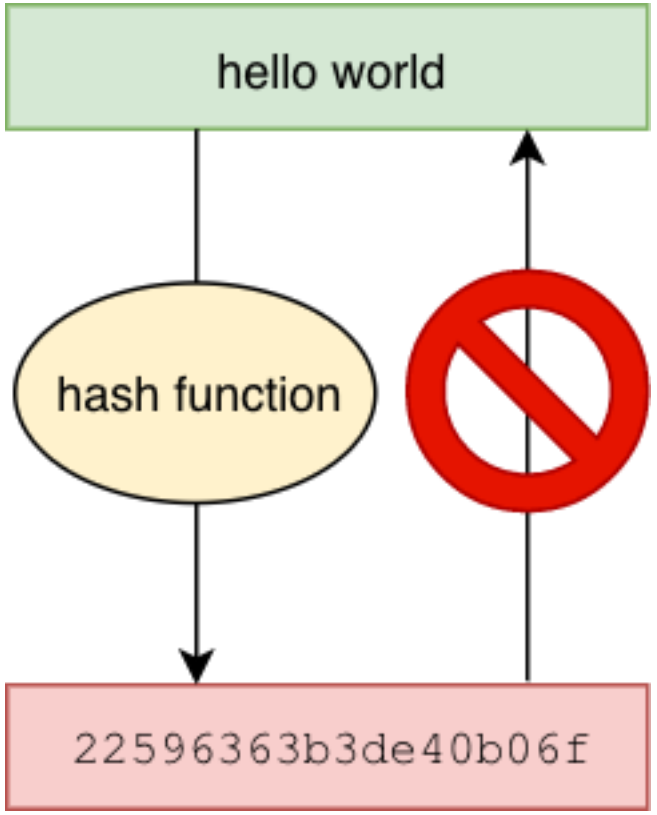
Digital Signature

- Signatures assure the receiver
  - of the sender's identity, and
  - that the contents were not altered in transit.
- Independent of encryption.
- Conceptually, it's asymmetric encryption in reverse.
  - Only the sender can sign.
  - Anyone (with the public key) can verify.
- More details to follow in a later slide.



Hash

- A hashing function take any data as input and produces a fixed length result in a repeatable way.
- The term hash is a
  - **noun** – the output of the function, aka **digest**
  - **verb** – the act of applying the function
- Qualities of a "good" hash function
  - difficult to reverse
  - produces very different outputs for inputs that differ even slightly

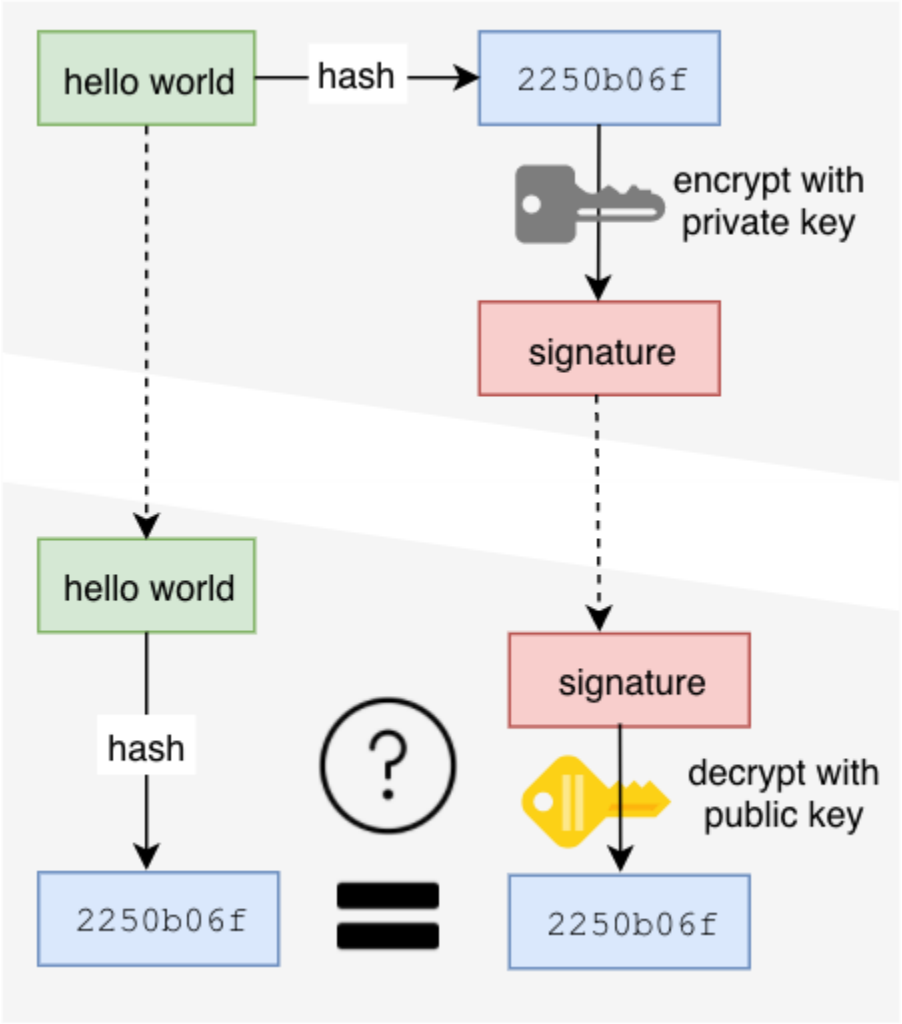


Digital Signatures – More Detail

- Asymmetric encryption and hash functions are used to implement digital signatures.

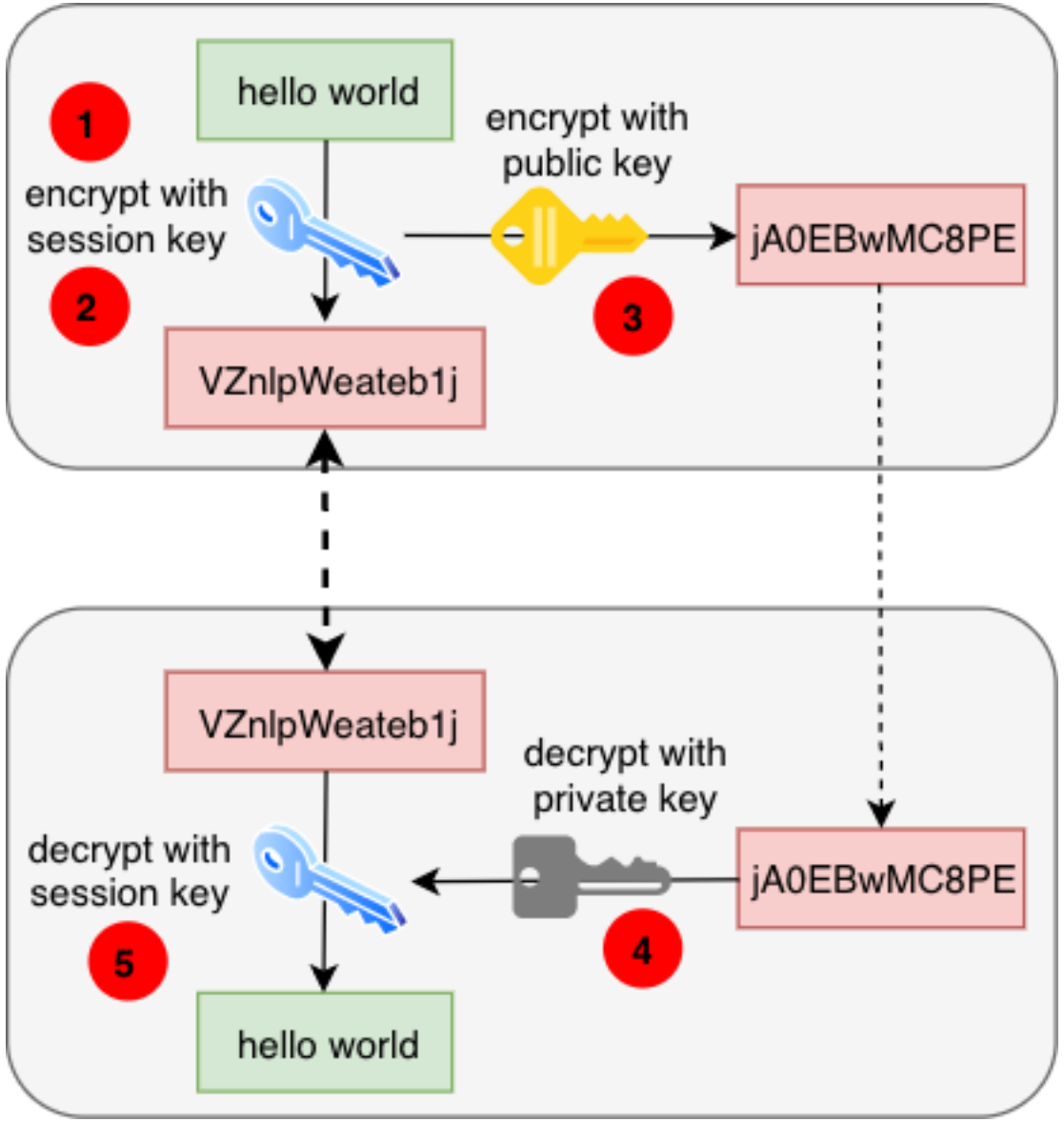
- On the sender side:
  - The content is hashed.
  - The hash is encrypted with the private key.
  - The content and signature are sent.

- On the receiver side:
  - A new hash is created from the content.
  - Original hash is decrypted from signature.
  - Signature is valid if the two hashes are equal.



Session Keys

- Asymmetric encryption between two parties is almost always undertaken via symmetric session keys.
    1. Sender creates a session key.
    2. The message is encrypted with the session key.
    3. The session key is encrypted with the recipient's public key.
    4. The recipient decrypts the session key using the private key.
    5. The session key decrypts the message.
- Additional encrypted exchanges may continue to use the symmetric key for the lifetime of the session. (See bold dotted two-way connection).

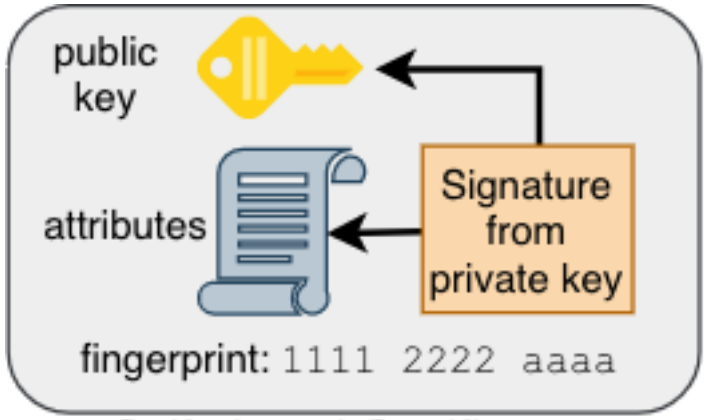


Certificates

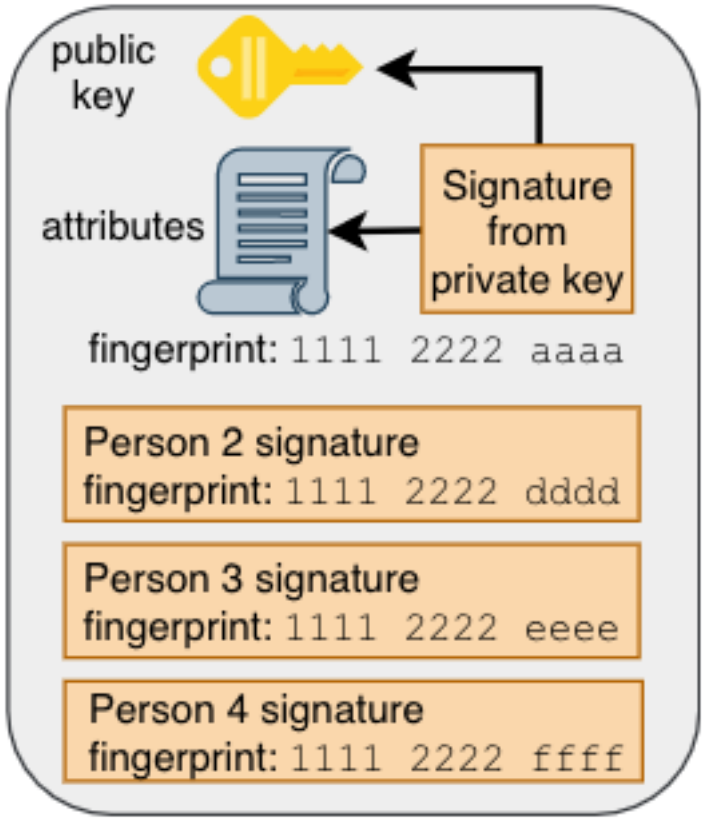
- The term *certificate* is often used interchangeably with the term *public key*.

- A certificate is a public key wrapper.
  - The public key contains the mathematical information required for crypto operations with the private key.
  - The certificate also contains attributes such as identifiers and expiration.
  - The public key and attributes are signed. The certificate contains this signature.

- Whose private key created the signature?
  - If the only signature comes from the key owner, the certificate is *self-signed*. The only way to verify is to contact owner about the fingerprint.
  - If others have signed the public key, you may already trust one of these other parties, sparing you the task of independent verification.



Self-signed Certificate

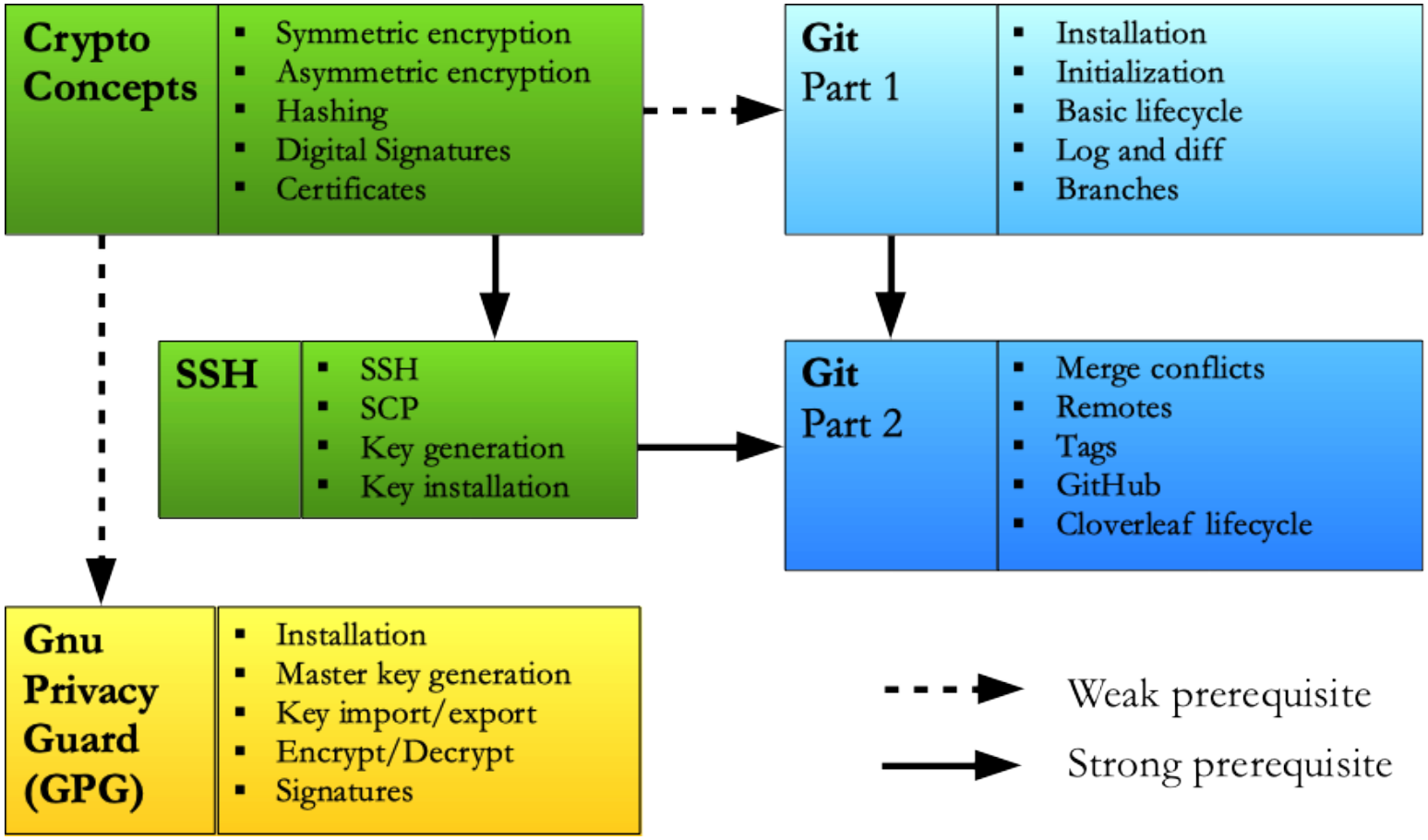


Certificate

Certificate Signers

- TLS (formerly SSL)
  - Certificates generally represent organizations.
  - A trusted signer is called a CA – Certificate Authority.
  - SSL applications have a **trust store**.
    - The trust store contains root and/or self-signed certificates.
    - The web browser is such an example.
      - Initial set of CAs determined by vendor
      - User may add or remove CAs
    - Each SSL application has a trust store config.
  - Signers form a hierarchy.
    - Top of chain is the root.
    - May contain several intermediate signers.
    - Customer's of a CA have their certificates signed by an intermediate signer.
- SSH – All certificates are self-signed.
- OpenPGP
  - A certificate generally represents an individual.
  - Signers are peers of the individual.
  - Signers form a web of trust rather than a hierarchy.
  - Verification of certificates requires some knowledge on the part of the user (hence this presentation).
- Uses
  - **TLS** typically encrypts TCP connections. Once connection is established, all traffic is encrypted.
  - **SSH** is also connection encryption.
  - **OpenPGP** typically encrypts fixed content. The result may be sent over insecure transports such as email or FTP.

Road Map


Crypto Concepts
Symmetric encryption
Asymmetric encryption
Hashing
Digital Signatures
Certificates
Git Part 1
Installation
Initialization
Basic lifecycle
Log and diff
Branches
SSH
SSH
SCP
Key generation
Key installation
Git Part 2
Merge conflicts
Remotes
Tags
GitHub
Cloverleaf lifecycle
Gnu Privacy Guard (GPG)
Installation
Master key generation
Key import/export
Encrypt/Decrypt
Signatures
Weak prerequisite
Strong prerequisite