Wallet Security

35th Chaos Communication Congreß, Leipzig, Germany

Stephan Verbücheln

December 28, 2018

My Background

Professional Background

- Diplominformatiker (eq. master's degree in CS)
- Security Analyst (cnlab security ag, Switzerland)

Blockchain-related work

- Research on zero-knowledge proofs and Zerocoin (predecessor of predecessor of Zcash)
- Research on ECDSA attacks in the context of Bitcoin
- Blockchain protocol architect (Trestor, Canada/India)
- Blockchain security review (Æternity, Liechtenstein)
- Wallet security review (several)

Agenda

- Recap of Bitcoin and ECDSA
- Wallets
- Common attacks
- Kleptographic attack
- Conclusions

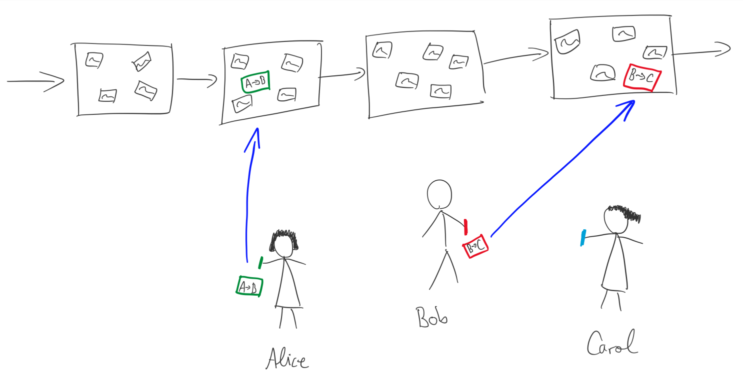
Bitcoin

Bitcoin

Public ledger for transactions.

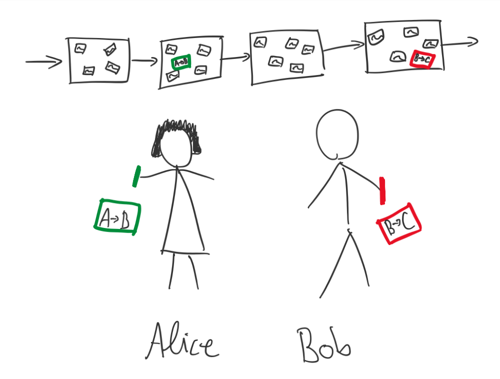
- Users have public-private key pairs.
- Transactions are signed with private keys.
- Transactions are published on the blockchain.

The Network



- Alice creates a transaction to Bob and broadcasts it
- Miners collect transactions and include them
- Eventually one miner mines a block with the transaction
- Bob waits for a few blocks to confirm

Transactions



Alice creates the transaction as follows:

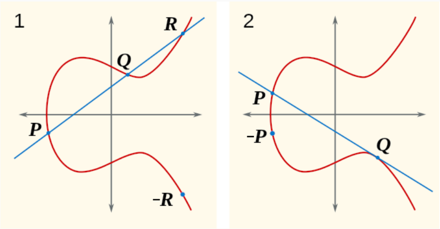
- Alice selects a coin that she owns
- She writes a transaction to Bob's address
- She signs the transaction with her private key

ECDSA

ECDSA

- Elliptic-Curve Digital Signature Algorithm
- Evolution of related algorithms:
  - Diffie-Hellman (discrete logarithm modulo p)
  - ElGamal signature
  - Schnorr signature
  - Digital Signature Algorithm (DSA)
- Why elliptic curves?
  - RSA and DH have no future
  - 4096-bit keys are not significantly stronger than 2048-bit keys

Elliptic Curves



Point $P = (p_1, p_2)$ on a curve $y^2 = x^3 + ax + b$

1. Addition:
   - $P + Q + R := 0$
   - $P + Q = -R$
2. Scalar multiplication:
   - $P + Q + Q := 0$
   - $2Q := Q + Q = -P$

Easy to compute: $Q = dG$. Hard to compute the reverse.

Signatures

Point G (order n), hash function $\mathcal{H}$.

Private key d, public key $Q = dG$.

sign(m)

1. Pick random nonce $k < n$.
2. Compute $R = (r_1, r_2) = kG$.
3. Compute $r = r_1 \mod n$.
4. Compute $s = k^{-1}(\mathcal{H}(m) + dr) \mod n$.
5. Return (r, s).

verify$(m, (r, s))$

1. Compute $R' = (r'_1, r'_2) = s^{-1}\mathcal{H}(m)G + s^{-1}rQ$.
2. Compute $r' = r'_1 \mod n$.
3. Test whether $r = r'$.

Properties

Point G (order n), hash function $\mathcal{H}$.
Private key d, public key $Q = dG$.

$\text{sign}(m)$

1. Pick random nonce $k < n$.
2. Compute $R = (r_1, r_2) = kG$.
3. Compute $r = r_1 \mod n$.
4. Compute $s = k^{-1}(\mathcal{H}(m) + dr) \mod n$.
5. Return (r, s).

Observation:

- With k you can compute $d = (\mathcal{H}(m) - sk)r^{-1} \mod n$.
- This means that k has to be kept secret.

Wallets

Wallets

- Secure storage of secret keys
- Signing of transactions
- Backup plans

Sofware Wallets vs. Hardware Tokens

Types of wallets

- Software
  - Can be used on desktop, laptop, phone, server
  - Flexible, full user control
  - Keys might be exposed through attacks on the host
- Hardware
  - Dedicated hardware tokens
  - Keys cannot be accessed from the host
  - How does the token know what it is signing?
- Paper
  - Backup only

Hardware Key Storage

Properties

- Keys are imported or generated in hardware
- Keys can be flagged non-exportable
- Signatures are performed inside the hardware module
- But note: Privileged access enables to *use* the keys.

Downsides

- Bugs cannot be easily fixed
- Implementation cannot be validated by the user

Examples

- Server HSM (hardware security module)
- TPM in business laptops
- Smartphone

Common Problems and Attacks

- Secrets leaked via network
  - Backdoors
  - Malware
- Secrets stored insecurely
  - Hardware theft
  - Malware
- Predictable random numbers
  - Attacker guesses private keys
  - Collision (re-use) of nonce k

Cryptographic Backdoors

Backdoor'd random number generators

- Famous example: Dual_EC_DRBG

Malicious wallet with cryptographic backdoor

- The nonce k is generated by a backdoor'd RNG.
- Attacker scans all transactions on the blockchain
- …and uses his backdoor to compute the secret key d.

Kleptograms

Kleptograms

- Term first coined by Adam Young and Moti Yung in 1997.

Notation

- Lower-case letters ($a, t, k_1, \dots$) for numbers
- Capital letters ($G, A, \dots$) for points on the curve
- Greek letters ($\alpha, \beta, \omega, \dots$) for constants
- $\mathfrak{R}(\cdot)$ is a random-number generator

Predictable R

RNG $\mathfrak{R}$. Generating two subsequent choices k_1, k_2:

First round.

1. Pick random $k_1 < n$.
2. Store k_1.
3. Output k_1 and $R_1 = k_1 G$.

Note that R_1 will be part of the signature.

Second round.

1. Compute $k_2 = \mathfrak{R}(R_1)$.
2. Output k_2 and $R_2 = k_2 G$.

Extraction of k_2

Second round.

1. Compute $k_2 = \mathfrak{R}(R_1)$.
2. Output k_2 and $R_2 = k_2 G$.

Extraction of the (secret) value k_2:

1. Compute $k_2 = \mathfrak{R}(R_1)$

Observation:

- Anyone can compute $k_2 = \mathfrak{R}(R_1)$.
- Can we hide it?

Kleptogram in R_2

Attacker's key pair a and $A = aG$. RNG $\mathfrak{R}$. Generating two subsequent choices k_1, k_2:

First round.

1. Pick random $k_1 < n$.
2. Store k_1.
3. Output k_1 and $R_1 = k_1 G$.

Second round.

1. Pick random bit $t \in \{0, 1\}$.
2. Compute $Z = (k_1 - \omega t)G + (-\alpha k_1 - \beta)A$.
3. Compute $k_2 = \mathfrak{R}(Z)$.
4. Output k_2 and $R_2 = k_2 G$.

Extraction of k_2

Second round.

1. Pick random bit $t \in \{0, 1\}$.
2. Compute $Z = (k_1 - \omega t)G \; + \; (-\alpha k_1 - \beta)A$.
3. Compute $k_2 = \mathfrak{R}(Z)$.
4. Output k_2 and $R_2 = k_2 G$.

Extraction of the (secret) value k_2:

1. Compute $T = \alpha R_1 + \beta G$.
2. Compute $Z_1 = R_1 - aT$.
3. If $R_2 = \mathfrak{R}(Z_1)G$ then output $k_2 = \mathfrak{R}(Z_1)$.
4. Compute $Z_2 = Z_1 - \omega G$.
5. If $R_2 = \mathfrak{R}(Z_2)G$ then output $k_2 = \mathfrak{R}(Z_2)$.

Attack on Wallets

Attack Scenario

Preparation

- The attacker backdoors a popular wallet.

Patience

- Victims create transactions with the wallet.
- Following the Bitcoin protocol, transactions are published on the blockchain.

Harvest

- The attacker scans the blockchain for signatures generated by the same key.
- The attacker uses his secret to derive private keys.

Attack Properties

- Only reused keys are vulnerable.
  - Using the same key multiple times is common in Bitcoin.
  - The same key might be used in one transaction.
- But note, that some applications require key reuse.
- Also note that in deterministic wallets, the attacker might derive further keys.

Notes

- The attack is independent from the consensus in Bitcoin.
- It applies to other blockchains with similar signatures.
- The backdoor also applies to other protocols using ECDSA.

Conclusions

Conclusions

What does this mean for users?

- Keys can be leaked through transactions.
- No side channel required.
- Cannot be detected by traffic analysis.

What to do now?

- Be very careful choosing your wallet.
- Even in an isolated environment.
- For some applications, transparency might be more important than tampering resistance.

Contact and References

Contact: verbuecheln@posteo.de

PGP fingerprint: 41D6 B8D2 A422 5DF1 AEE1 EA63 6035 4259 0A3C 7C62

References

- IETF, *RFC 6979: Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)*, 2013
- Adam Young, Moti Yung, *The Prevalence of Kleptographic Attacks on Discrete-Log based Cryptosystems*, CRYPTO '97
- Stephan Verbücheln, *How Perfect Offline Wallets Can Still Leak Bitcoin Private Keys*, MCIS 2015

Pictures

- Curve diagram based on work by Wikipedia/SuperManu (GNU FDL)