WiP: Towards a Secure SECP256K1 for Crypto Wallets: Hardware Architecture and Implementation

Joel Poncha Lemayian, Ghyslain Gagnon, Kaiwen Zhang, and Pascal Giard

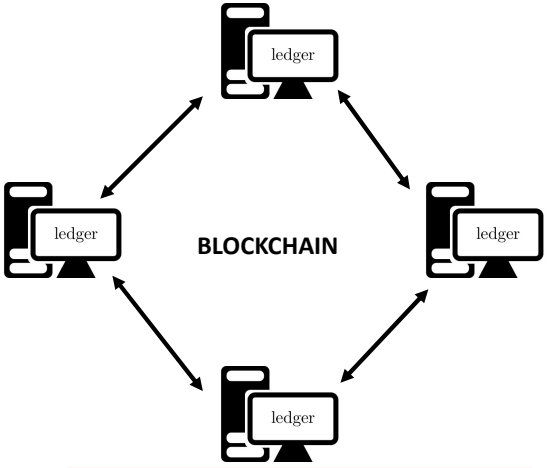
Laboratoire de Communications et d'Intégration de la MicroÉlectronique (LaCIME),
École de technologie supérieure (ÉTS)

November 2^{nd}, 2024

Outline

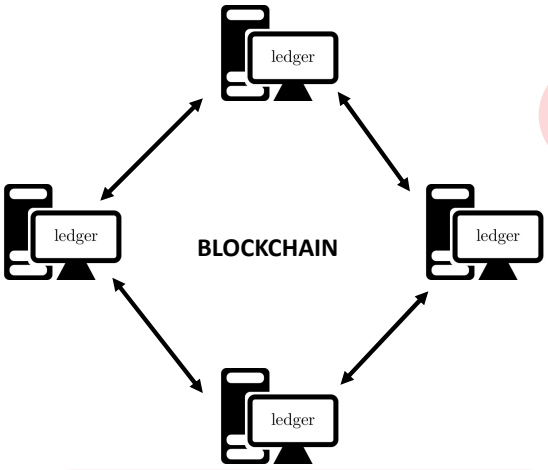
- Background
- Motivation
- Objectives
- Contributions
- Results
- Conclusion

Blockchain Technology



Decentralized shared ledger of transactions

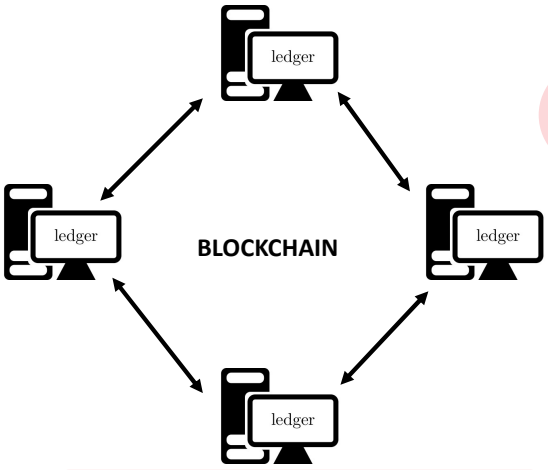
Blockchain Technology



Applications: Supply chain management,
Internet of things,
Cryptocurrency (Crypto)

Decentralized shared ledger of transactions

Blockchain Technology

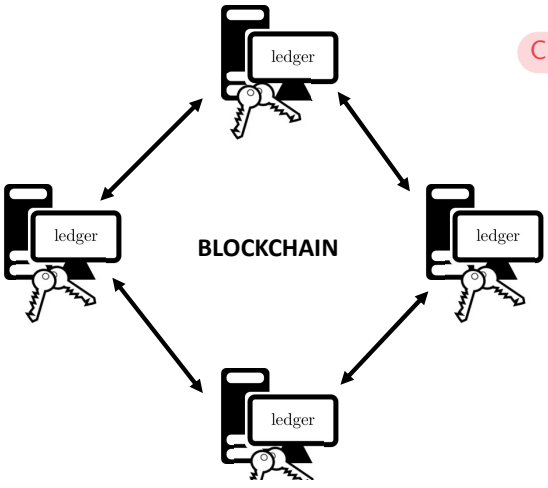


Decentralized shared ledger of transactions

Applications: Supply chain management, Internet of things, Cryptocurrency (Crypto)

Examples: Bitcoin, Ethereum, etc.

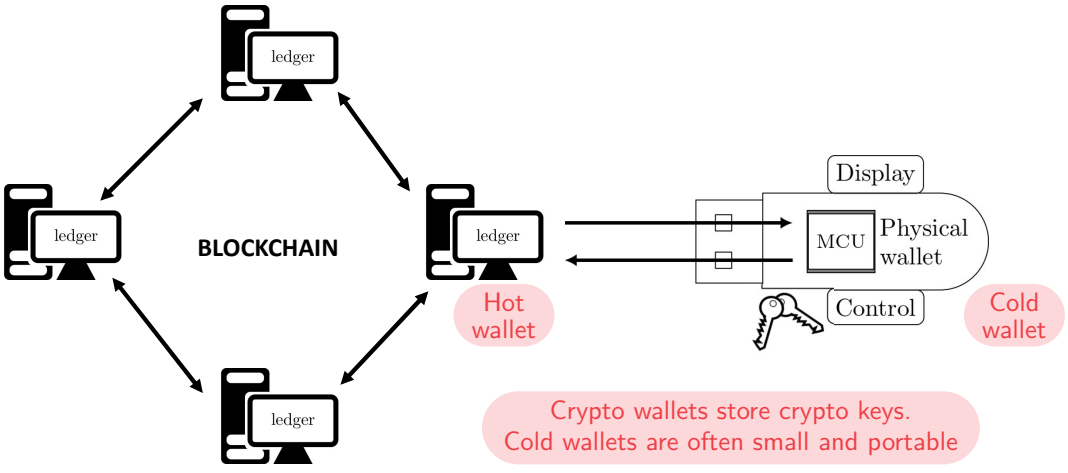
Blockchain Technology



Crypto key provide ownership of digital assets.

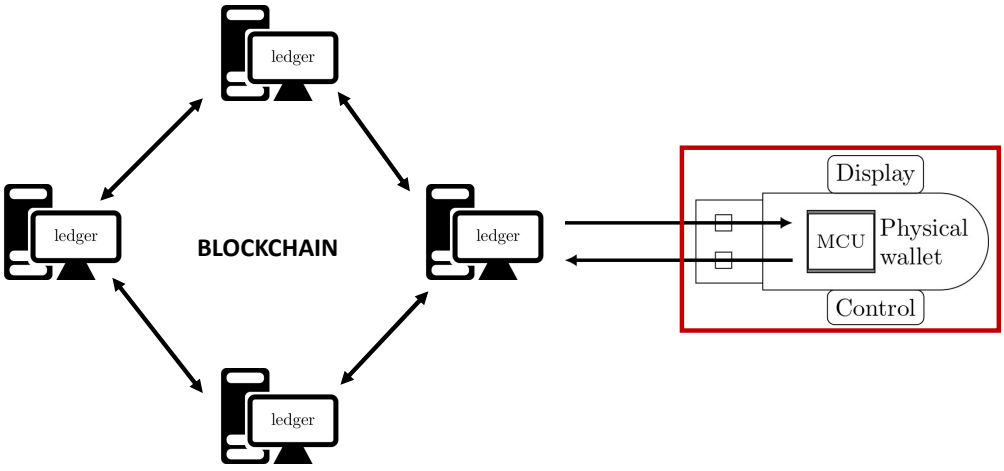
There are public and private keys.

Blockchain Technology

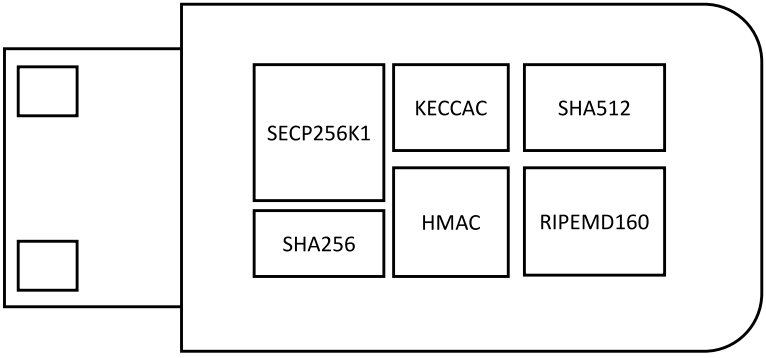


Crypto wallets store crypto keys.
Cold wallets are often small and portable

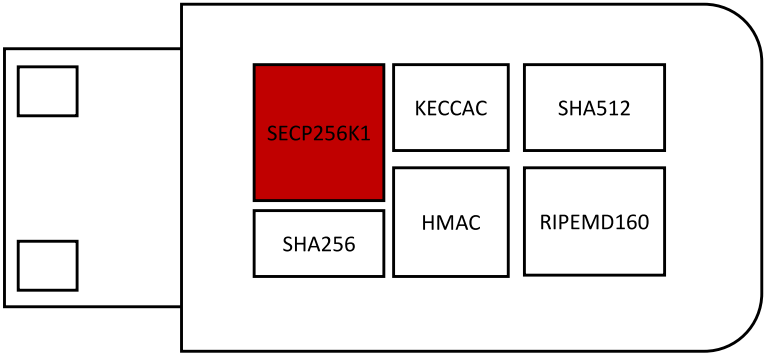
Blockchain Technology



Bitcoin and Ethereum wallets

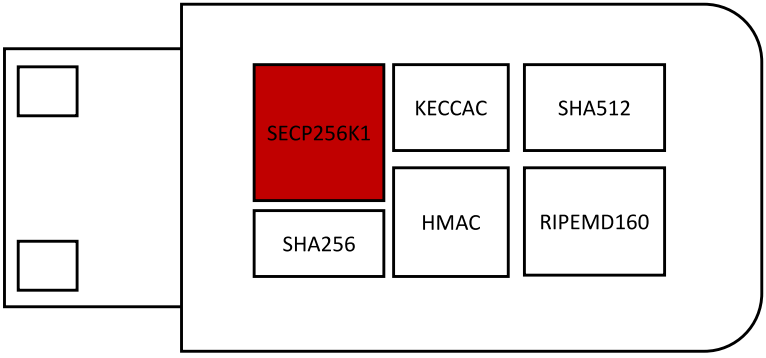


Bitcoin and Ethereum wallets



Computes a public key given a private key

Bitcoin and Ethereum wallets



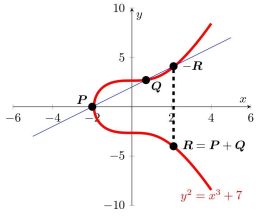
Various attacks targeted Elliptic curve cryptography (ECC) algorithm.

SECP256K1

SECP256K1

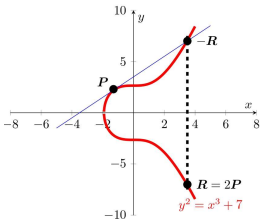
Elliptic curve point addition (ECPA)

R = P + Q



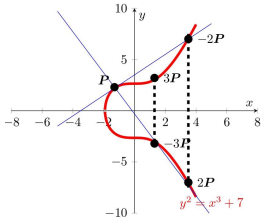
Elliptic curve point doubling (ECPD)

R = 2P



Elliptic curve point multiplication (ECPM)

R = *kP = $\sum_{i=1}^{k} P$**



Key vulnerability in Montgomery ladder ECPM

Algorithm 1 Montgomery Ladder

Input: $\boldsymbol{P} \in (x, y, z), \boldsymbol{m} = (m_{t-1}, \ldots, m_0)$ with $m_{t-1} = 1$
Output: $\boldsymbol{R} = \boldsymbol{mP}$

Initialisation:
1: $\boldsymbol{R_0} \leftarrow \boldsymbol{P}$
2: $\boldsymbol{R_1} \leftarrow 2\boldsymbol{P}$
Loop Process:
3: **for** $i = t - 2$ to 0 **do**
4: **if** $m_i = 1$ **then**
5: $\boldsymbol{R_0} \leftarrow \boldsymbol{R_0} + \boldsymbol{R_1}$
6: $\boldsymbol{R_1} \leftarrow 2\boldsymbol{R_1}$
7: **else**
8: $\boldsymbol{R_1} \leftarrow \boldsymbol{R_0} + \boldsymbol{R_1}$
9: $\boldsymbol{R_0} \leftarrow 2\boldsymbol{R_0}$
10: **end if**
11: **end for**
12: **return** $\boldsymbol{R_0}$

Private key bit = 1

Private key bit = 0

Power consumption pattern and execution time discrepancy

Objectives

- To secure SECP256K1 against side-channel analysis (SCA) attack.
  - Complete addition equation
  - Temporary registers
  - Parallel operations
- To minimize resources utilized by SECP256K1.
  - Efficiently reusing modules

Use equations to perform ECPA

Algorithm 2 Equations for complete, projective point addition for SECP256K1

Input: $\boldsymbol{P} = (X_1, Y_1, Z_1), \boldsymbol{Q} = (X_2, Y_2, Z_2)$ on $E : Y^2Z = X^3 + bZ^3$ and $b_3 = 3 \cdot b$.
Output: $(X_3, Y_3, Z_3) = \boldsymbol{P} + \boldsymbol{Q}$;

1: $t_0 \leftarrow X_1 \cdot X_2$
2: $t_1 \leftarrow Y_1 \cdot Y_2$
3: $t_2 \leftarrow Z_1 \cdot Z_2$
4: $t_3 \leftarrow X_1 + Y_1$
5: $t_4 \leftarrow X_2 + Y_2$
6: $t_3 \leftarrow t_3 \cdot t_4$
7: $t_4 \leftarrow t_0 + t_1$
8: $t_3 \leftarrow t_3 - t_4$
9: $t_4 \leftarrow Y_1 + Z_1$
10: $X_3 \leftarrow Y_2 + Z_2$
11: $t_4 \leftarrow t_4 \cdot X_3$
12: $X_3 \leftarrow t_1 + t_2$
13: $t_4 \leftarrow t_4 - X_3$
14: $X_3 \leftarrow X_1 + Z_1$
15: $Y_3 \leftarrow X_2 + Z_2$
16: $X_3 \leftarrow X_3 \cdot Y_3$
17: $Y_3 \leftarrow t_0 + t_2$
18: $Y_3 \leftarrow X_3 - Y_3$
19: $X_3 \leftarrow t_0 + t_0$
20: $t_0 \leftarrow X_3 + t_0$
21: $t_2 \leftarrow b_3 \cdot t_2$
22: $Z_3 \leftarrow t_1 + t_2$
23: $t_1 \leftarrow t_1 - t_2$
24: $Y_3 \leftarrow b_3 \cdot Y_3$
25: $X_3 \leftarrow t_4 \cdot Y_3$
26: $t_2 \leftarrow t_3 \cdot t_1$
27: $X_3 \leftarrow t_2 - X_3$
28: $Y_3 \leftarrow Y_3 \cdot t_0$
29: $t_1 \leftarrow t_1 \cdot Z_3$
30: $Y_3 \leftarrow t_1 + Y_3$
31: $t_0 \leftarrow t_0 \cdot t_3$
32: $Z_3 \leftarrow Z_3 \cdot t_4$
33: $Z_3 \leftarrow Z_3 + t_0$

Avoid the branching caused by SECP256K1 EC addition operation

Use temporary registers in ECPM

Algorithm 3 Montgomery Ladder Algorithm with Temporary Registers

Input: $\boldsymbol{P} \in (x, y, z), \boldsymbol{m} = (m_{t-1}, \ldots, m_0)$ with $m_{t-1} = 1$
Output: $\boldsymbol{R} = \boldsymbol{mP}$

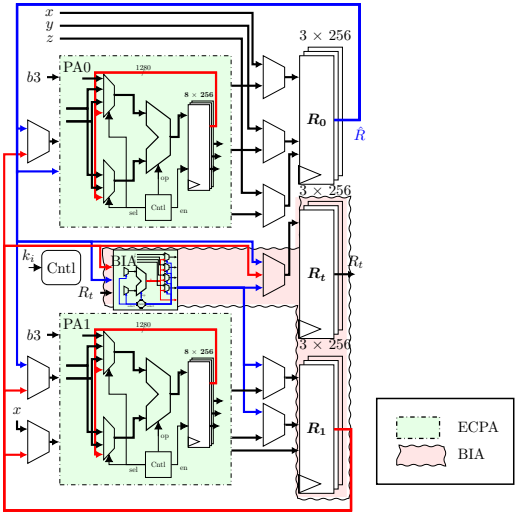
Initialisation:
1: $\boldsymbol{R_0} \leftarrow \boldsymbol{P}$
2: $\boldsymbol{R_1} \leftarrow 2\boldsymbol{P}$
Loop Process:
3: **for** $i = t - 2$ to 0 **do**
4: **if** $m_i = 1$ **then**
5: $\boldsymbol{R_0} \leftarrow \boldsymbol{R_0} + \boldsymbol{R_1}$
6: $\boldsymbol{R_1} \leftarrow 2\boldsymbol{R_1}$
7: $\boldsymbol{R_t} \leftarrow 2\boldsymbol{R_0}$
8: **else**
9: $\boldsymbol{R_1} \leftarrow \boldsymbol{R_0} + \boldsymbol{R_1}$
10: $\boldsymbol{R_0} \leftarrow 2\boldsymbol{R_0}$
11: $\boldsymbol{R_t} \leftarrow 2\boldsymbol{R_1}$
12: **end if**
13: **end for**
14: **return** $\boldsymbol{R_0}$

Private key bit = 1

Private key bit = 0

Both branches perform addition and doubling of the same registers

Use parallel operation in hardware implementation



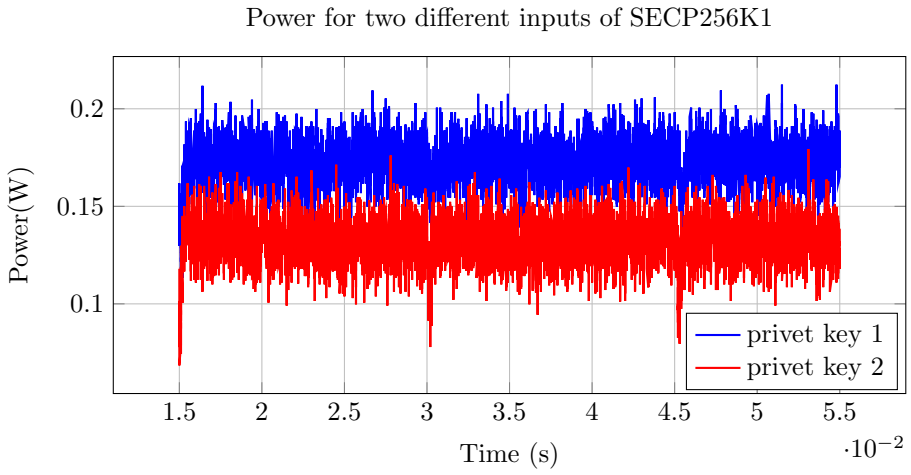
1. ECPM done in projective coordinates.
2. Binary inversion done at the end.
3. ECPA is done with two modules in parallel.
4. Registers reused to achieve minimum area.

SECP256K1 implementation results

Work	Platform	Area				Frequency	Latency		Throughput[a]
		kLUT	DSP	RAM (kbits)	Registers	(MHz)	(ms)	(kCC)	(kbps)
This work	**Zynq-US**	**21**	**0**	**0**	**13 881**	**250**	**7.58**	**1 895**	**34**
This work	**Artix-7**	**24**	**0**	**0**	**13 385**	**90**	**21**	**1 895**	**12**
Mehrabi et al.[1]	Virtex-7	47	560	0	29 742	125	0.25	N/A	N/A
Asif et al.[2]	Virtex-7	19	1 036	828	N/A	87	0.73	63	351
Islam et al.[3]	Virtex-7	36	N/A	N/A	N/A	178	1.48	2630	173
Romel et al.[4]	Virtex-7	52	0	N/A	15 263	122	0.54	66	476
Arunachalam et al.[5]	Virtex-5	33	N/A	N/A	N/A	192	1.21	232	212
Roy et al.[6]	Virtex-5	40	0	N/A	N/A	43	0.60	26	1 667
Asif et al.[7]	Virtex-7	97	2799	7 452	N/A	73	2.96	216	1 816

[a] Throughput is estimated by authors as (Frequency ÷ CC) × 256.

Power side channel analysis



MSE = 0.001840 => No significant difference

Conclusion

- Temporary registers and parallel operation used to mitigate SCA.
- MSE is small, suggesting protection against differential power analysis.
- Proposed architecture uses few resources.
- Future: Hardware architecture for a crypto wallet.

Thank you!

Refferences

1. Mohamad Ali Mehrabi, Christophe Doche, and Alireza Jolfaei. Elliptic curve cryptography point multiplication core for hardware security module.
2. Shahzad Asif, Md Selim Hossain, Yinan Kong, and Wadood Abdul. A fully RNS based ECC processor.
3. Md Mainul Islam, Md Selim Hossain, Moh Khalid Hasan, Md Shahjalal, and Yeong Min Jang. FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field.
4. Md Ashraful Islam Romel, Md Rafiul Islam, and Fathun Karim Fattah. FPGA implementation of elliptic curve point multiplication for a 256-bit processor on nist prime field.
5. Kamaraj Arunachalam and Marichamy Perumalsamy. FPGA implementation of time-area-efficient elliptic curve cryptography for entity authentication.
6. Sujoy Sinha Roy, Debdeep Mukhopadhyay, and West Bengal. Implementation of PSEC-KEM (secp256r1 and secp256k1) on hardware and software platforms final project report.
7. Shahzad Asif, Md Selim Hossain, and Yinan Kong. High-throughput multi-key elliptic curve cryptosystem based on residue number system.