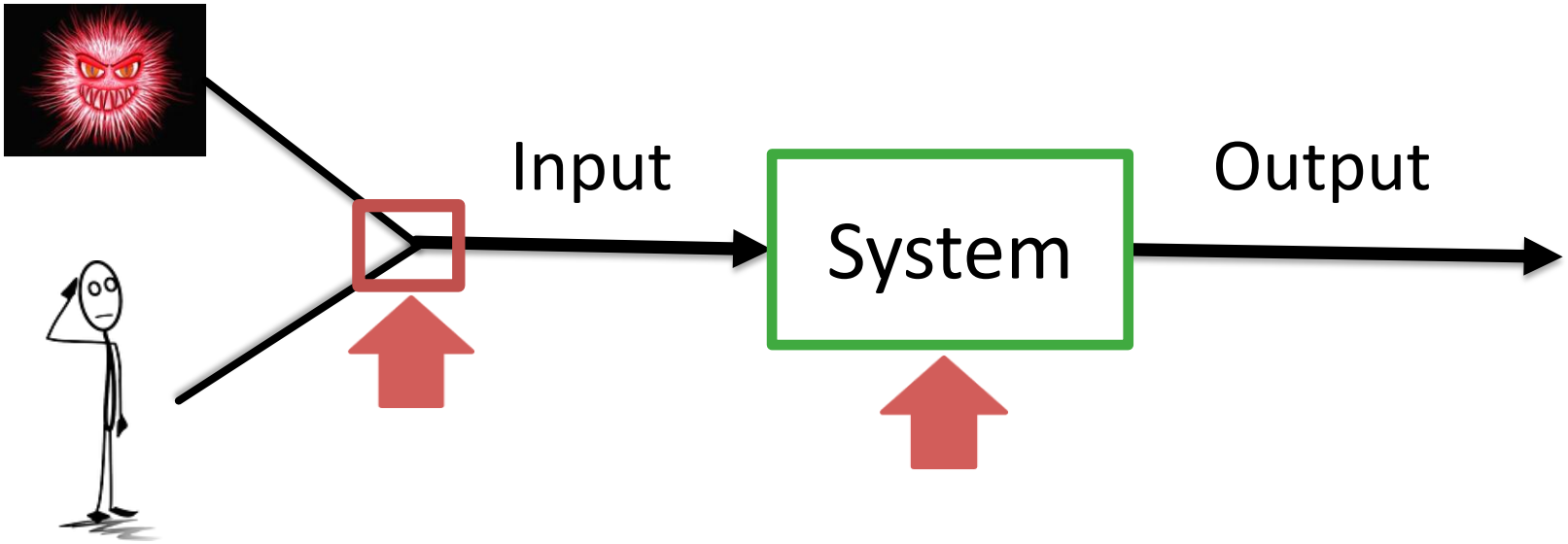
Smart-Guard: Defending User Input from Malware

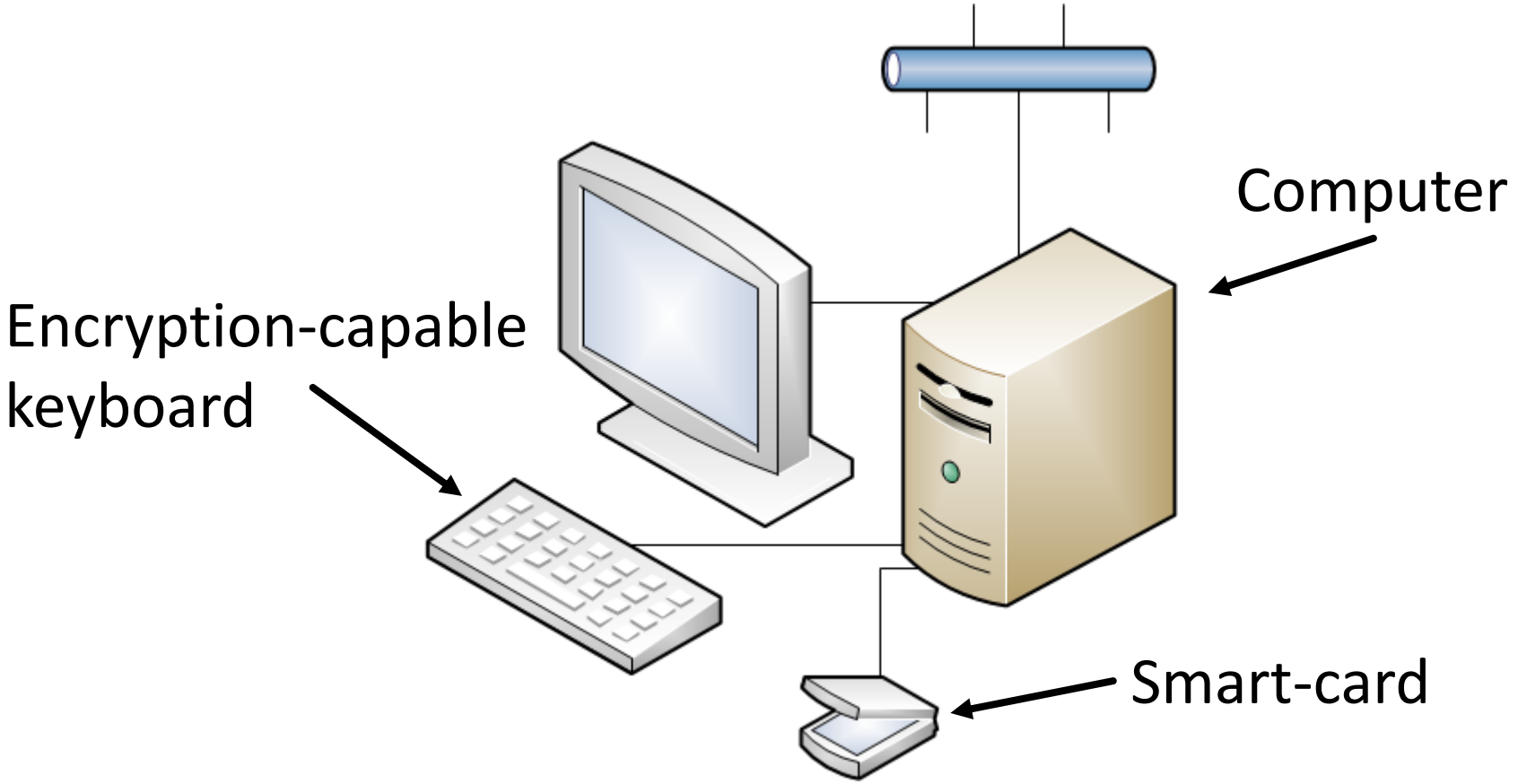
Michael Denzel, Alessandro Bruni, Mark Ryan
University of Birmingham

2016-07-21

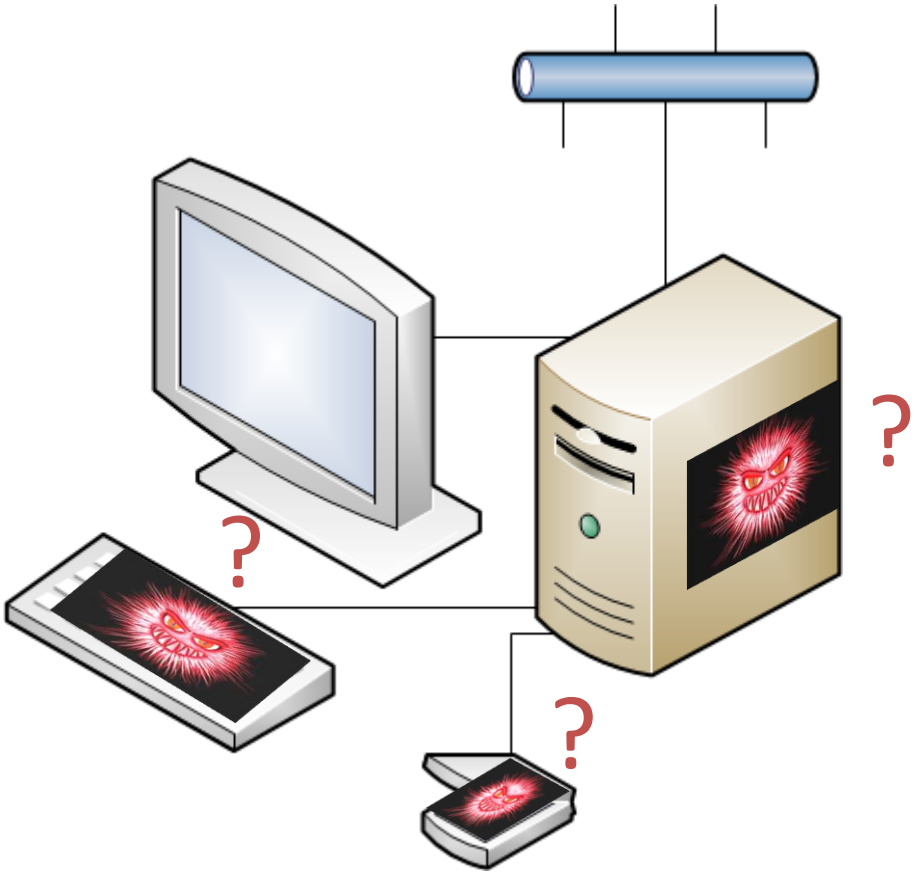
Problem



Smart-Guard

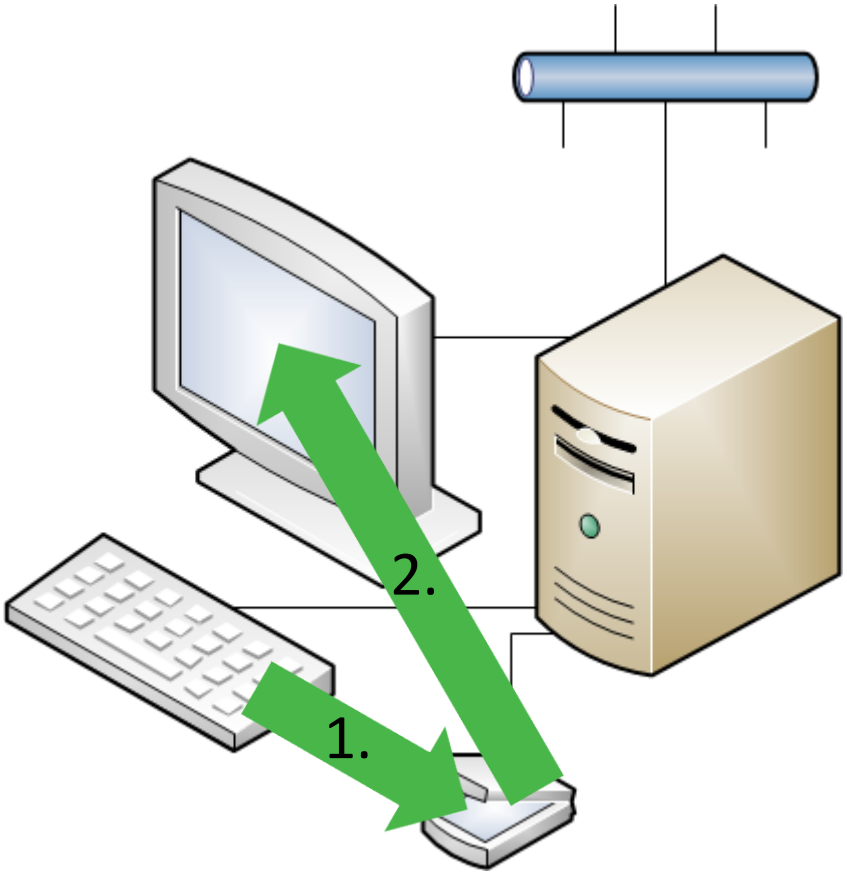


Flexible Trust Assumptions

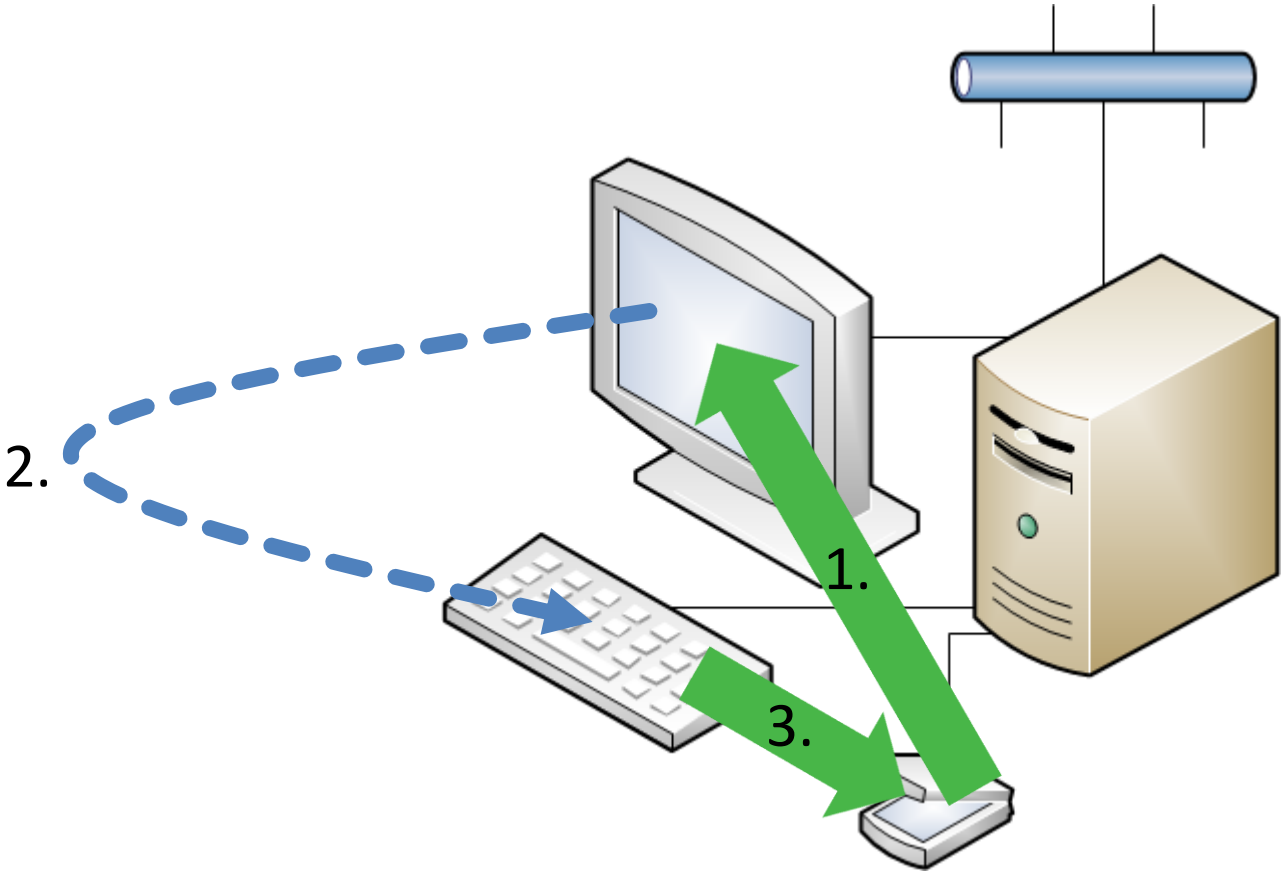


Distribute Trust!

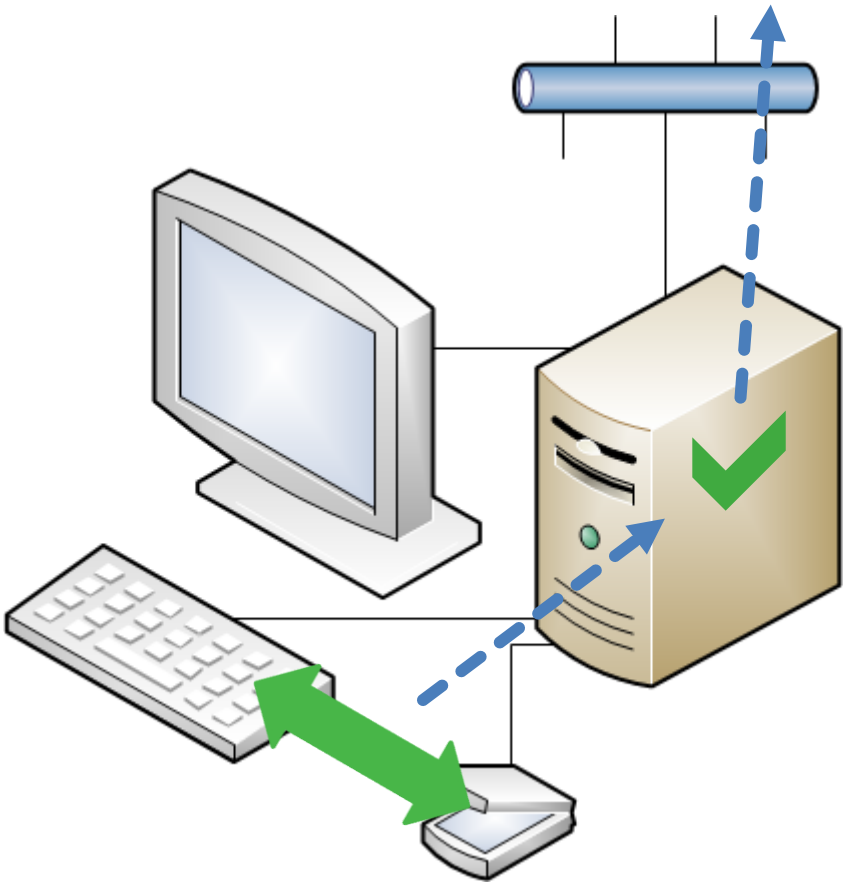
Smart-Guard: Input



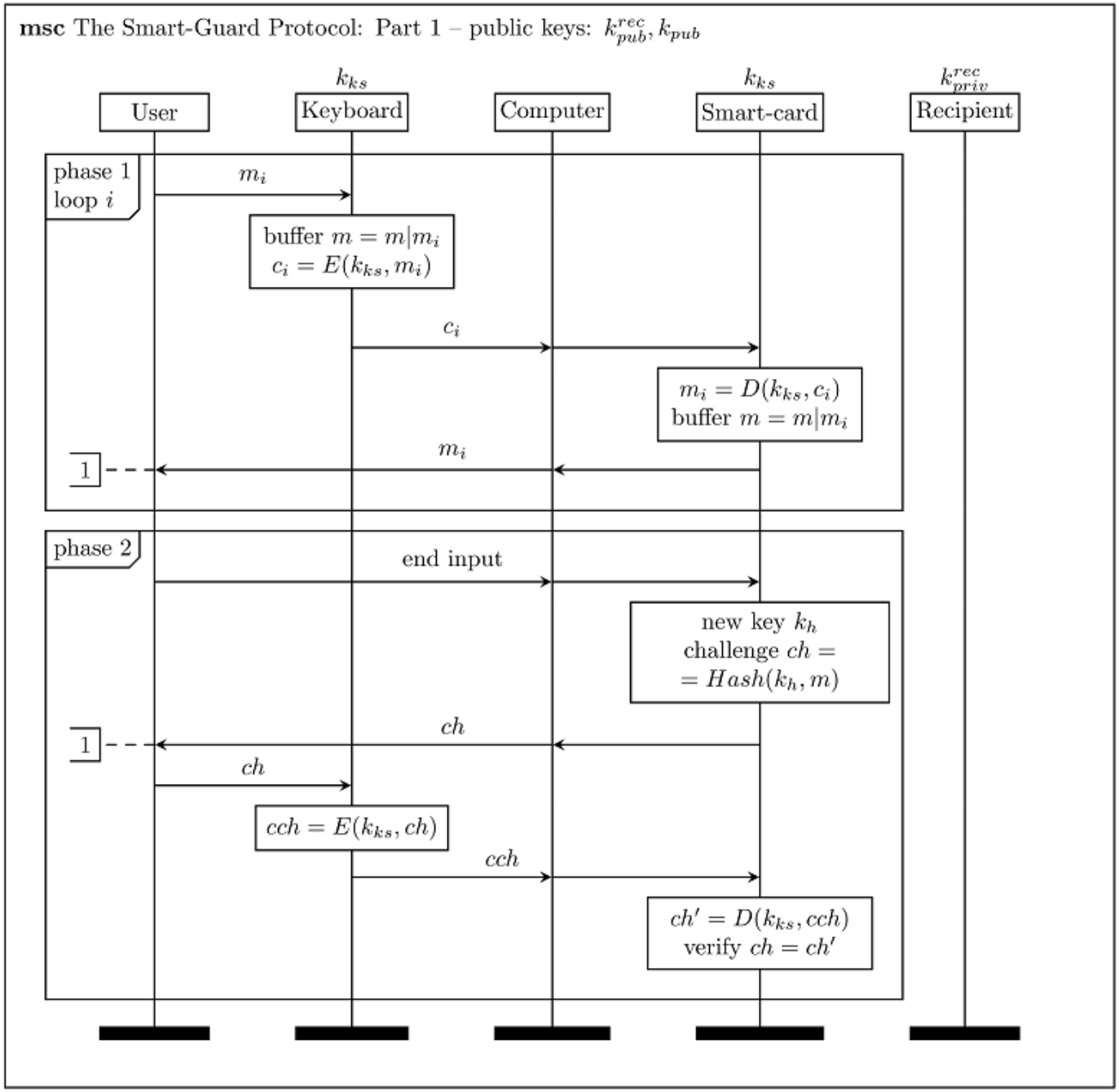
Smart-Guard: Challenge-Response

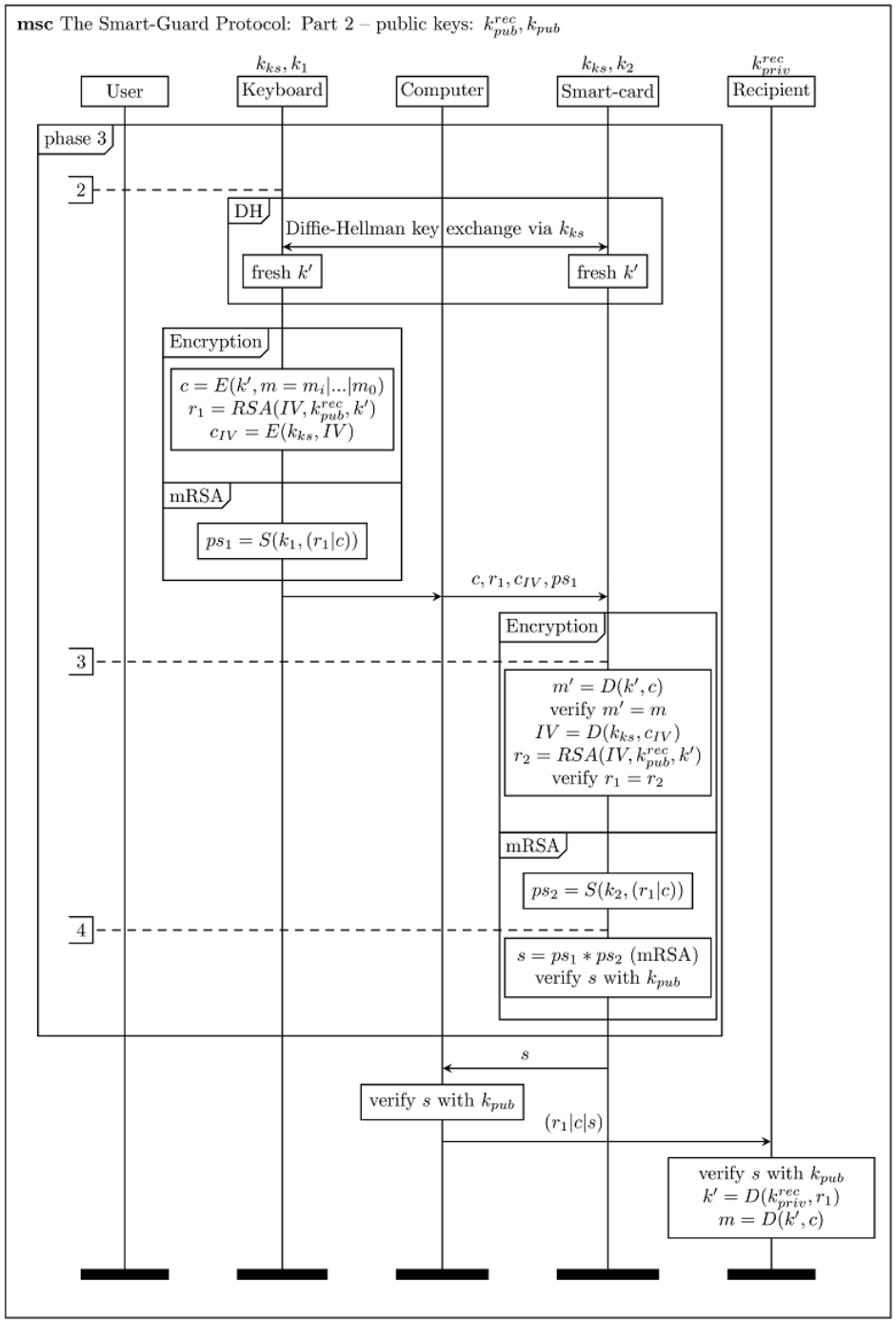


Smart-Guard: Encryption



DH → RSA → mRSA [9]


msc The Smart-Guard Protocol: Part 1 – public keys: k_{pub}^{rec}, k_{pub}
User
Keyboard
k_{ks}
Computer
Smart-card
k_{ks}
Recipient
k_{priv}^{rec}
phase 1
loop i
m_i
buffer $m = m|m_i$
$c_i = E(k_{ks}, m_i)$
c_i
$m_i = D(k_{ks}, c_i)$
buffer $m = m|m_i$
m_i
1
phase 2
end input
new key k_h
challenge $ch =$
$= Hash(k_h, m)$
ch
1
ch
$cch = E(k_{ks}, ch)$
cch
$ch' = D(k_{ks}, cch)$
verify $ch = ch'$


msc The Smart-Guard Protocol: Part 2 – public keys: k_{pub}^{rec}, k_{pub}
User
k_{ks}, k_1
Keyboard
Computer
k_{ks}, k_2
Smart-card
k_{priv}^{rec}
Recipient
phase 3
2
DH
Diffie-Hellman key exchange via k_{ks}
fresh k'
fresh k'
Encryption
$c = E(k', m = m_i|...|m_0)$
$r_1 = RSA(IV, k_{pub}^{rec}, k')$
$c_{IV} = E(k_{ks}, IV)$
mRSA
$ps_1 = S(k_1, (r_1|c))$
c, r_1, c_{IV}, ps_1
Encryption
3
$m' = D(k', c)$
verify $m' = m$
$IV = D(k_{ks}, c_{IV})$
$r_2 = RSA(IV, k_{pub}^{rec}, k')$
verify $r_1 = r_2$
mRSA
$ps_2 = S(k_2, (r_1|c))$
4
$s = ps_1 * ps_2$ (mRSA)
verify s with k_{pub}
s
verify s with k_{pub}
$(r_1|c|s)$
verify s with k_{pub}
$k' = D(k_{priv}^{rec}, r_1)$
$m = D(k', c)$

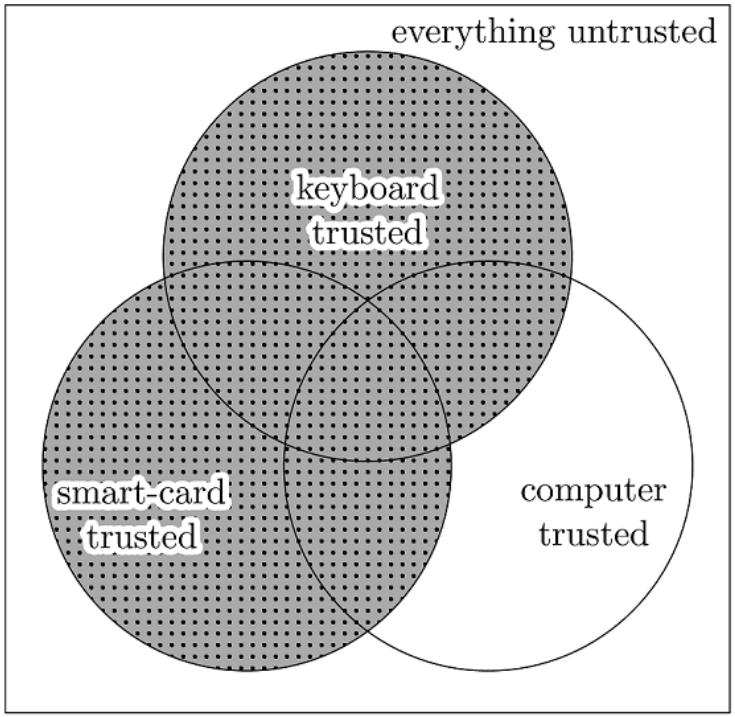
Theory

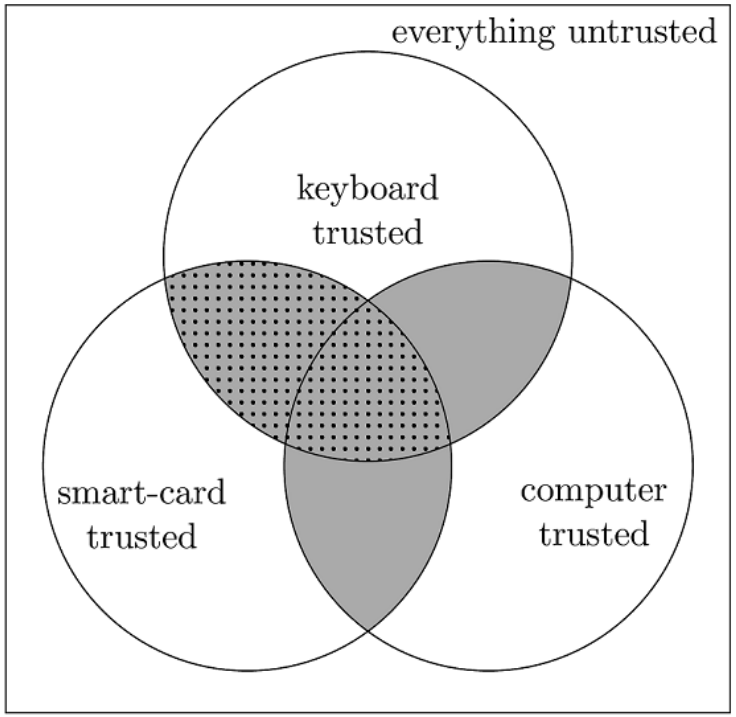
Multiple TCBs:

$$(TCB_1 \text{ secure}) \vee (TCB_2 \text{ secure}) \vee \ldots \Rightarrow \text{security}$$

Confidentiality:	Integrity:
$TCB_1 = \{\text{PC}, \text{smartcard}\}$	$TCB_1 = \{\text{keyboard}\}$
$TCB_2 = \{\text{smartcard}, \text{keyboard}\}$	$TCB_2 = \{\text{smartcard}\}$
$TCB_3 = \{\text{PC}, \text{keyboard}\}$	

Results





1. Integrity
2. Confidentiality

ProVerif proofs: https://github.com/mdenzel/smartguard

Comparison

Technique	Trusted Input	Trusted Output	Confidentiality	Integrity	Software attacks	Hardware attacks
BitE [2]	✔	~	✔	✔	~	
Bumpy [3]	✔	~	✔	✔	✔	
UTP [4]	✔			✔	✔	
DriverGuard [5]	✔	✔	✔	✔	~	
Key Scrambler [6]	✔		✔	✔	~	
Zhou et al. [7]	✔	✔	✔	✔	✔	
TrustZone [8]	✔	✔	✔	✔	~	
Smart-Guard [1]	✔	~	✔	✔	✔	✔

+ flexible TCB!

Summary

- Technique for Trusted Input
- A system can be (partly) compromised but secure!

Thank you!

[1] Michael Denzel, Alessandro Bruni and Mark Ryan: Smart-Guard: Defending User Input from Malware

[2] McCune, J.M., Perrig, A., Reiter, M.K.: Bump in the ether: A framework for securing sensitive user input

[3] McCune, J.M., Perrig, A., Reiter, M.K.: Safe passage for passwords and other sensitive data

[4] Filyanov, A., McCune, J.M., Sadeghiz, A.R., Winandy, M.: Uni-directional trusted path: Transaction confirmation on just one device

[5] Cheng, Y., Ding, X., Deng, R.H.: Driverguard: A fine-grained protection on I/O flows

[6] QFX software: https://www.qfxsoftware.com/ks-windows/how-it-works.htm

[7] Zhou, Z., Gligor, V.D., Newsome, J., McCune, J.M.: Building verifiable trusted path on commodity x86 computers

[8] ARM TrustZone: http://www.arm.com/products/processors/technologies/trustzone/index.php?tab=Hardware+Architecture

[9] Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certicates and security capabilities