White Box Remote Program Execution

Work in progress

Amir Herzberg
Haya Shulman
Bar Ilan university

Amitabh Saxena
Bruno Crispo
University of Trento

Talk Outline

- White Box Security vs. Black Box Security
- White-Box Remote Program Execution (WBRPE)
- Definitions
- WBRPE Robust Combiner
- Universal White Box RPE
- Further research and conclusions

Security/cryptography by obscurity

- Parties use secret scheme, keys
- Adversary observes traffic
  - Assumed not to know scheme – obscurity
- Standard assumption…
- Till Kerckhoff [1883]: only secret <u>keys</u>
  - Avoid `security by obscurity`
  - Attacker knows the algorithm (and code)

White Box vs. Black Box Security

- Kerckhoff: attacker knows all <u>but keys</u>
- Black-box security: keys kept in `trusted computing base` (TCB)
  - Available to (trusted) code, <u>not to attacker</u>
  - Tamper-proof hardware or trusted computer
  - Only <u>oracle access</u> (inputs / outputs)
- White-box security: keys encoded <u>in code</u>
  - Code generated by some <u>known process</u>
  - White-box security ≠ security by obscurity !!

Why is White-Box Interesting?

- Practical scenarios, e.g.:
  - Agents running in (untrusted) marketplace
  - Grid computing (using untrusted hosts)
  - DRM, Trusted Computing
    - Enforce organization policies
    - Protect user of insecure environment (Windows?)
- Theoretical interest
  - Can we establish (prove) white-box security?

Establishing White-Box Security

- Can we establish white-box security?
- [Barak et al.]: no `obfuscator`
  - Transform code to secure white-box code
- Our goal: explore approaches that <u>may</u> work
  - I.e. provide white-box security
- First, look at <u>black-box</u> security

Establishing Black-Box Security

- Three approaches:
  - Direct proof of security
  - Failure to cryptanalyze
  - Proof of reduction
- Direct proof seems best, but may not be feasible:
  - Provably-secure encryption for computationally-unbounded adversary ➔ |key|=|data| [Shannon]
    - Often not practical
  - Provably-secure encryption for poly-time adversaries ➔ proof that P≠NP
    - Too much to ask for

Establishing Black-Box Security by Failure to Cryptanalyze

- 'I showed it to Coppersmith in the elevator and he didn't find an attack`
- Classical (pre-1977?) approach:
  - Designers, experts try to break system
  - No known break (in spite of usage)
- Modern approach: open process
  - Competition, challenges, prizes, reputation…
  - Another advantage of Kerckhoff's principle

Establishing Black-Box Security by Proof of Reduction

`To see that colored cycle stripping is decidable, we reduce it to the halting problem.' [from `invalid proof techniques`]

- All: rely on `failure to break` (what we reduce to)
- Reductions to number theoretic problems
  - El-Gamal, Cramer-Shoup encryption: reduction to DDH
    - DDH (Decisional Diffie-Hellman): variant of discrete-log
- Reduction to weaker primitive
  - E.g. strong One Way Function (OWF) from weak OWF
- Reductions to **building blocks**
  - Two types: practical/standard, vs. theoretical/weakest

Building Block: Goals

- Simple, well defined
- Robust combiners:
  - Reduction to two (or more) candidates
  - Encryption $E''_{k''}(E'_{k'}(m))$ is secure if either E'' <u>or</u> E' is secure
- Theoretical/weakest building blocks:
  - (Trapdoor, weak) One-way function
  - Not for real use (loss of efficiency, security)
- Practical/standard building blocks:
  - Easily, efficiently applicable for many tasks
  - Security established by failed efforts to cryptanalyze

Practical/Standard Building Blocks

- Block ciphers
  - Data Encryption Standard (DES)
  - Advanced Encryption Standard (AES)
- Cryptographic hash functions
  - Secure Hash Algorithm (SHA), …
- `Practical/standard` public key cryptography
  - Digital Signature Standard (DSA/DSS)
  - RSA with PKCS (v. 1.5, v. 2) encoding
- Widely used
- Security based on failed attempt to break

Establishing White-Box Security

Three approaches (cf. black-box):

1. Direct proof of security
   1. Not likely (considering not likely for black-box)
2. Failure to cryptanalyze / attack
   1. Practical products/designs (obfuscators, DRM systems)
      1. Problem: mostly proprietary and/or weak
   2. Theoretical designs (based on encrypted computation)
      1. Problem: efficiency, limited applicability
   3. Need to define **building block**, find candidate
      1. Cf. block cipher (and DES, AES) WBRPE
3. Proof of reduction
   1. To building block, weaker scheme, robust combiner

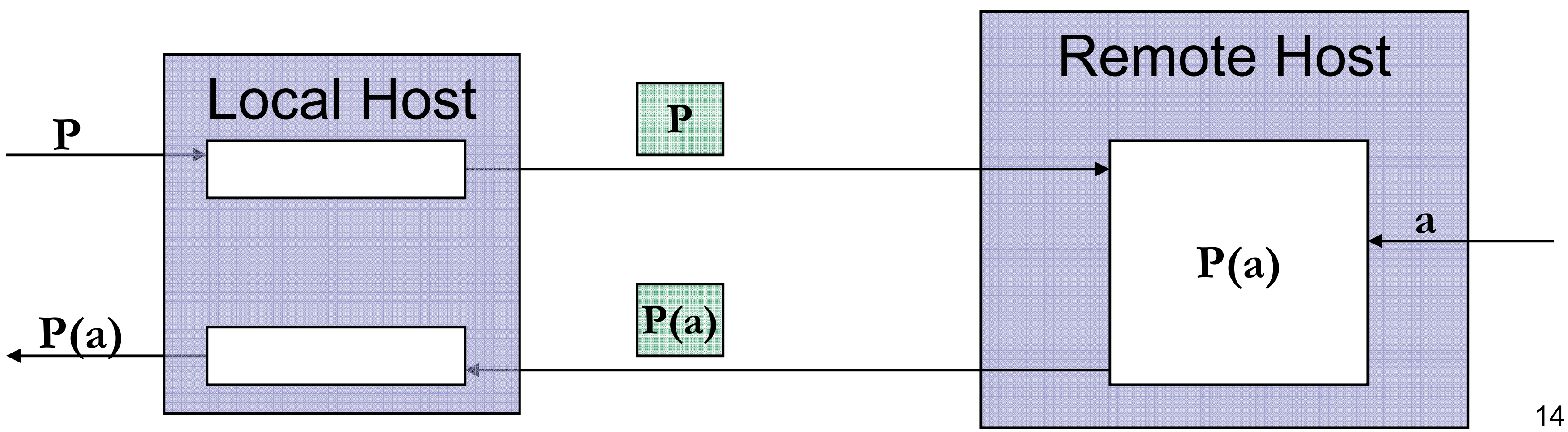
Our focus

Rest of this talk

- White-Box Remote Program Execution (WBRPE)
  - A building block
  - Basic yet useful; cf. to block cipher
- Definitions
- Reductions to establish security
  - Robust WBRPE combiner
    - Combined WBRPE scheme W''∘W' is secure, if either W'' or W' is secure
  - Universal reduction
    - Secure WBRPE for any program, given secure WBRPE for specific (keyed) program, UP_k
- Conclusions and future work

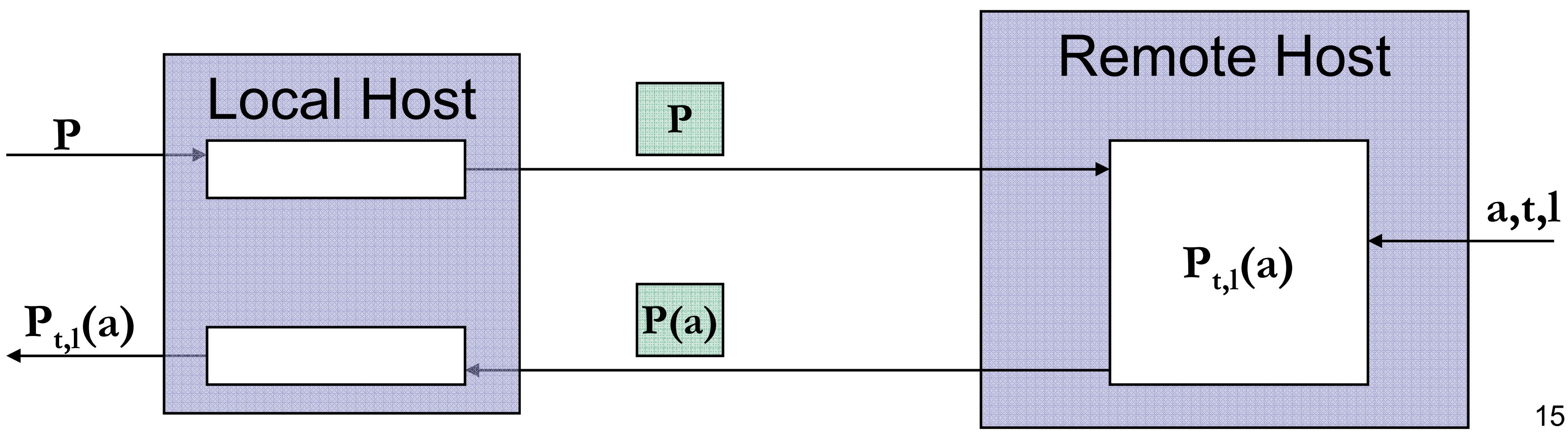
Remote Program Execution (RPE)

- Remote host has some remote data *a*
- Local host needs *P(a)* where *P* is some program (function)
  - P may include also local data, embedded inside
- Trivial solution: send *a* to local, compute *P(a)*
- Problems:
  - Data *a* may be private – not to be sent to local host
  - Sending data may be expensive



RPE: Efficiency Specifications

- Define max running time (t), output length (l)
- Reasonable overhead:
  - Communication bits < |P| + t + poly(*k*)
  - poly(*k*) computations at local host
  - t*poly(*k*) computations at remote host

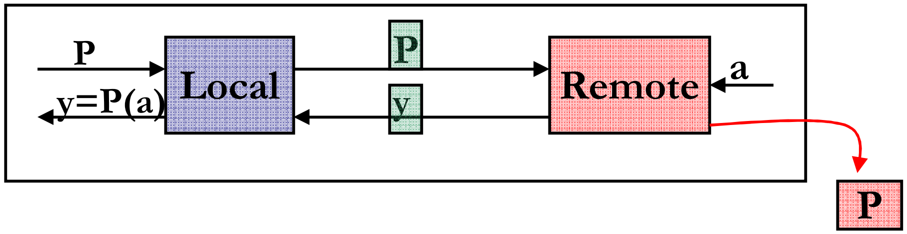


RPE Associated Security Threats

- Threats to local host
  - Program exposure (and of the embedded data)
  - Output forgery
- Threats to remote host
  - Exposure of remote input *a*
  - Execution of malware

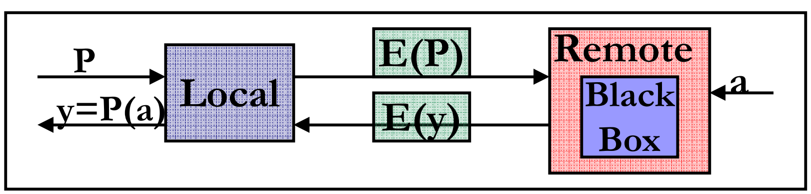
Program Exposure Threat

- Program P exposed
  - Program exposed → can be observed
  - Extraction of secret data embedded in P
    - cryptographic keys, credit card number, e-cash
- Program Privacy (indistinguishability of P)



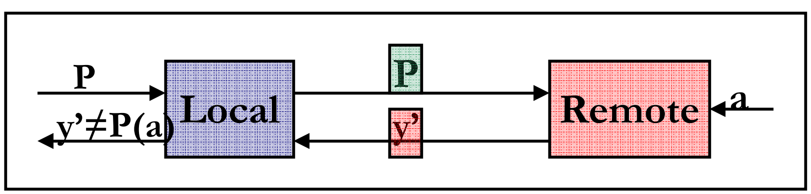
Program Privacy: Black Box Solution

- Achieving program privacy is easy with black box
- Local host encrypts program
- Remote host executes in black box
  - Decrypts, execution, encryption
  - Send to local host
- Local decrypts the result



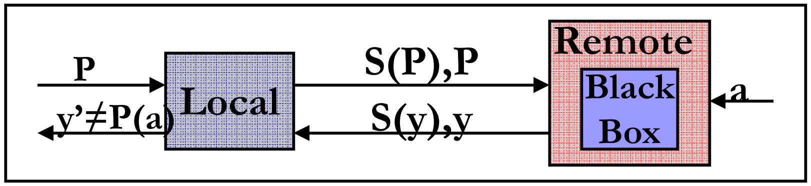
Forgery Threat

- Forgery of local output
- Unforgeability specification
  - Output y=⊥ for forgery
  - Output y≠⊥ only if (∃a), s.t. y=P(a)



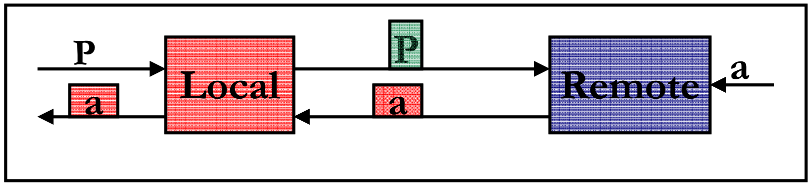
Forgery: Black Box Solution

- Sign program P sent to remote
- Sign output sent from remote
  - For confirming authenticity, origin and integrity
  - Trusted computing base, employ authentication techniques



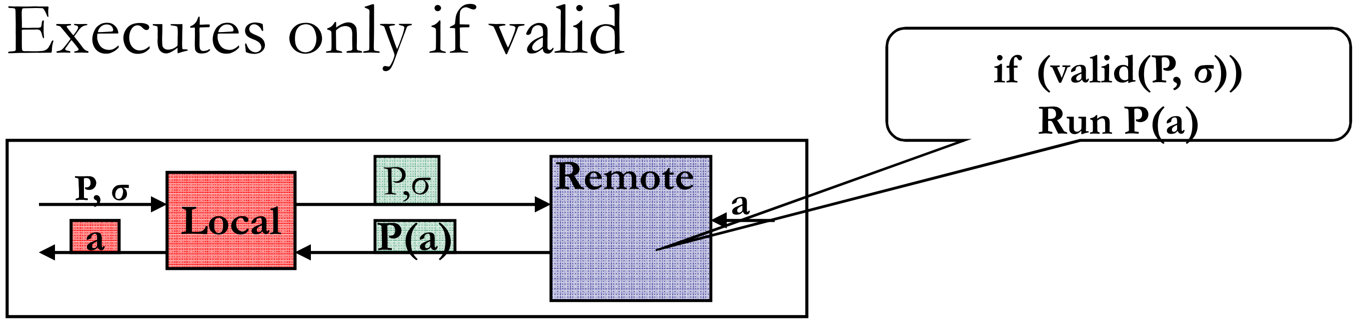
Remote Input Privacy Threat

- Local host chooses P, receives P(a)
- May choose P such that e.g. P(a)=a
  - Local host obtains the remote input
    - Or part of it
  - Remote input may be a private database



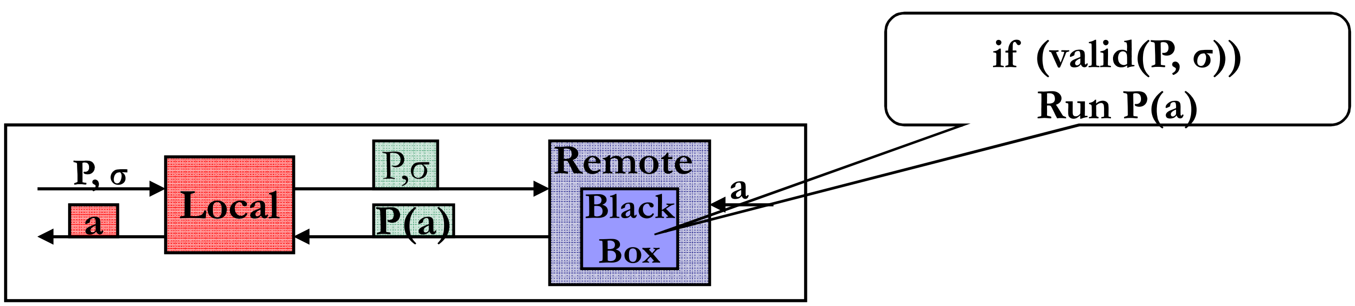
Remote Input Privacy: without program privacy

- Allow only valid P
  - Validation function
  - Validation parameter
- If no program privacy, then the validation is obtained trivially w/ o black box
  - The remote host validates the input program
  - Executes only if valid



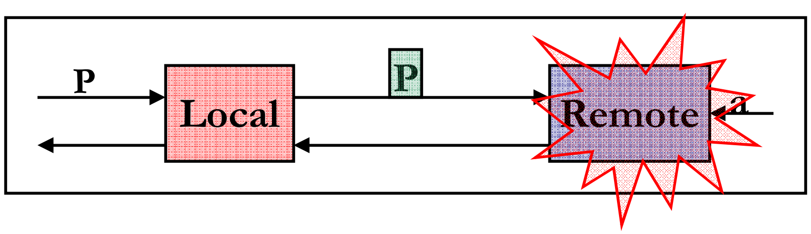
Remote Input Privacy, with program privacy: Black Box Solution:

- If program privacy is required: validate in black box
- Remote input semantic privacy
  - Why not indistinguishability?



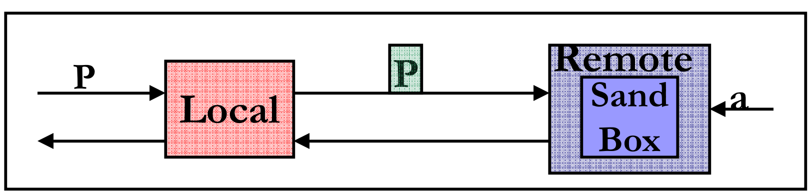
Remote Host Malware Threat, Remote Program Execution

- Damage host or its data
  - Malicious software – malware
    - computer viruses and trojan horses
- Protect remote host system from malware program P



Remote Host Security: Sandbox

- Protect remote host from malware program P
- Confine execution of program to sandbox
  - Access control
- Optionally also: signed code
  - For confirming authenticity, origin and integrity of P
  - Exploit trust relation with code originator
    - E.g. windows drivers

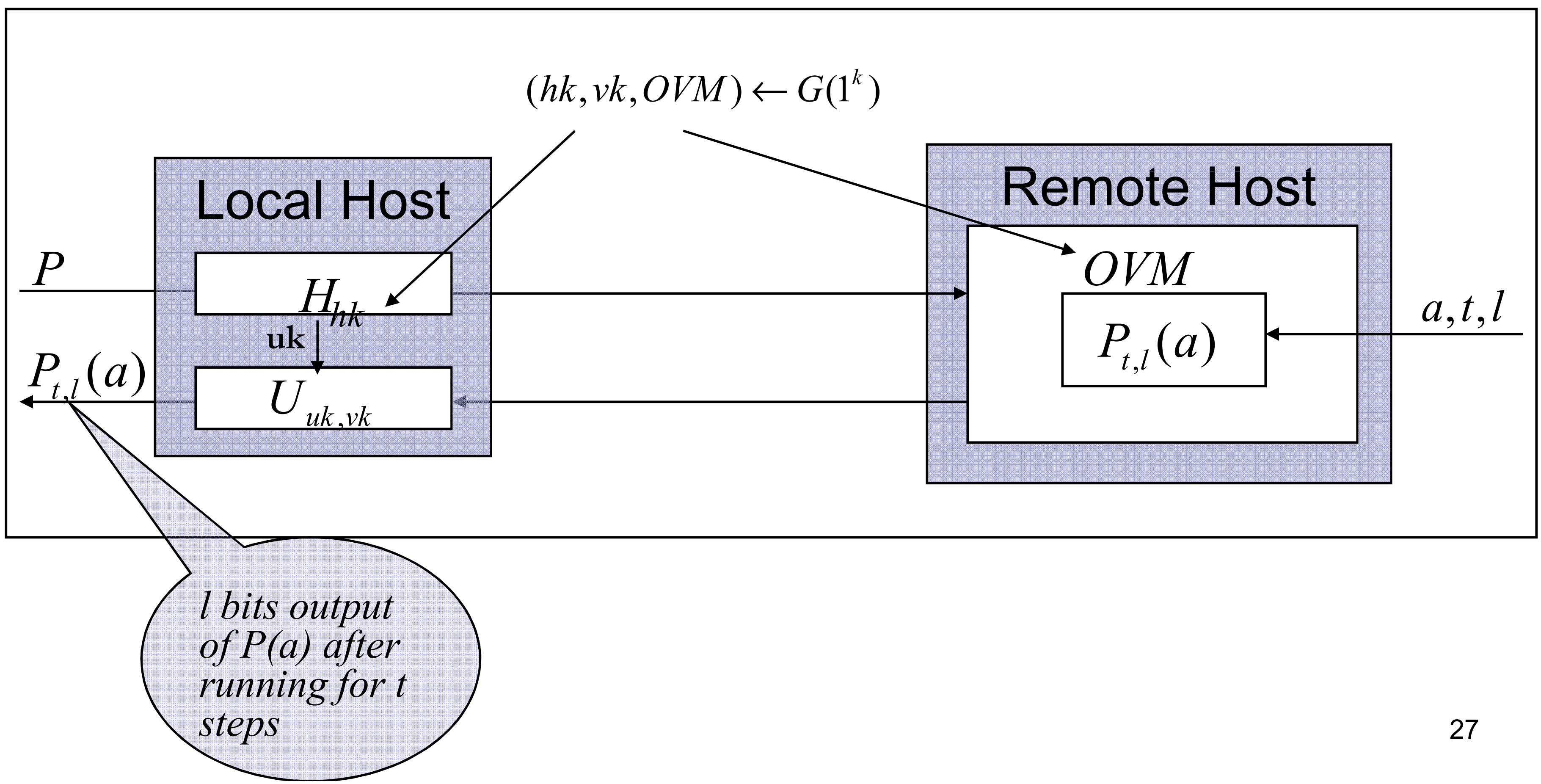


Talk Outline

- White Box Security vs. Black Box Security
- White-Box Remote Program Execution (WBRPE)
- Definitions
- WBRPE Robust Combiner
- Universal White Box RPE
- Further research and conclusions

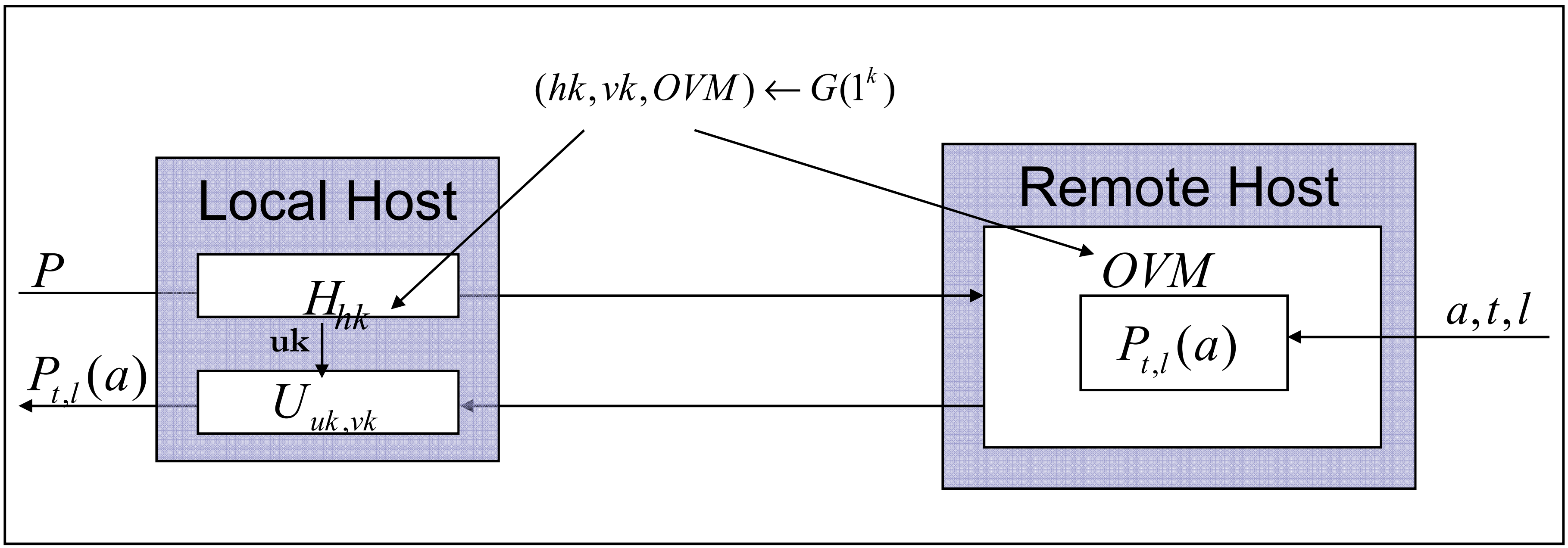
WBRPE: Definition

- *A WBRPE is a tuple of PPT algorithms <G,H,U>*

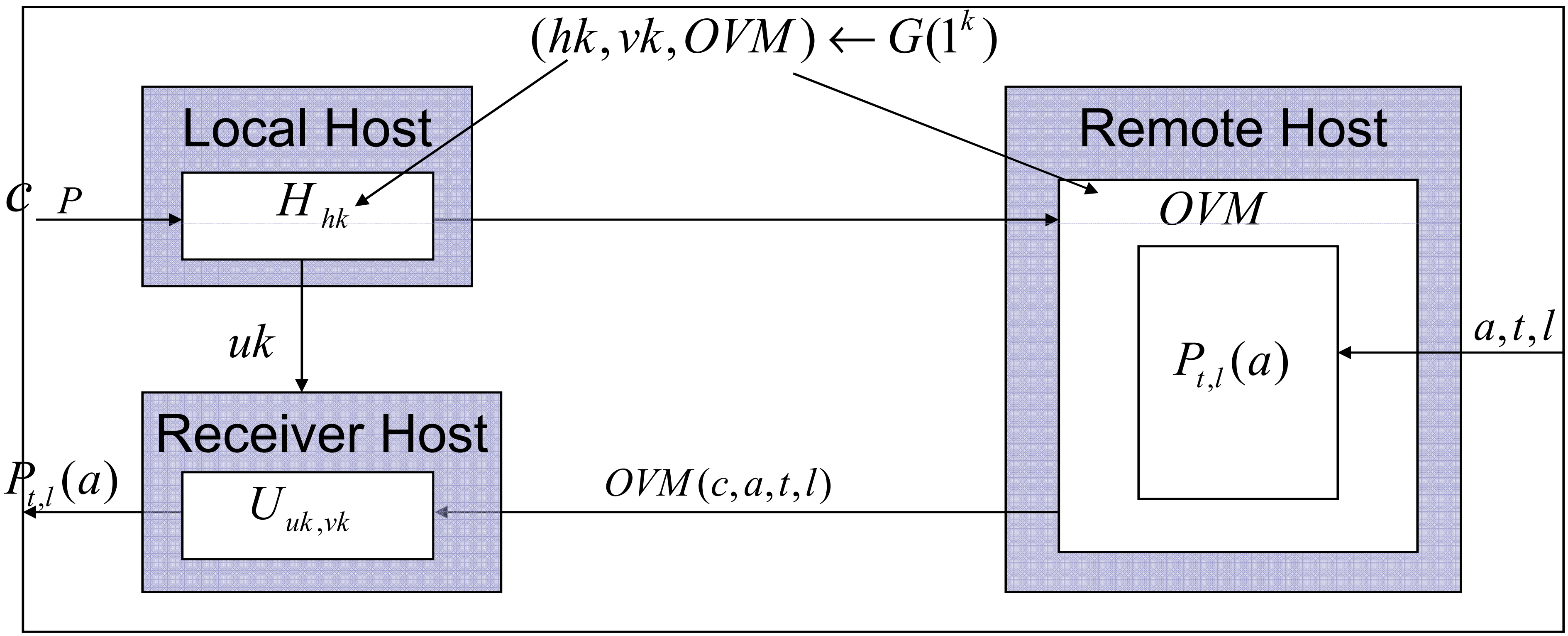


WBRPE: Restrictions

- Stateless
- No remote output

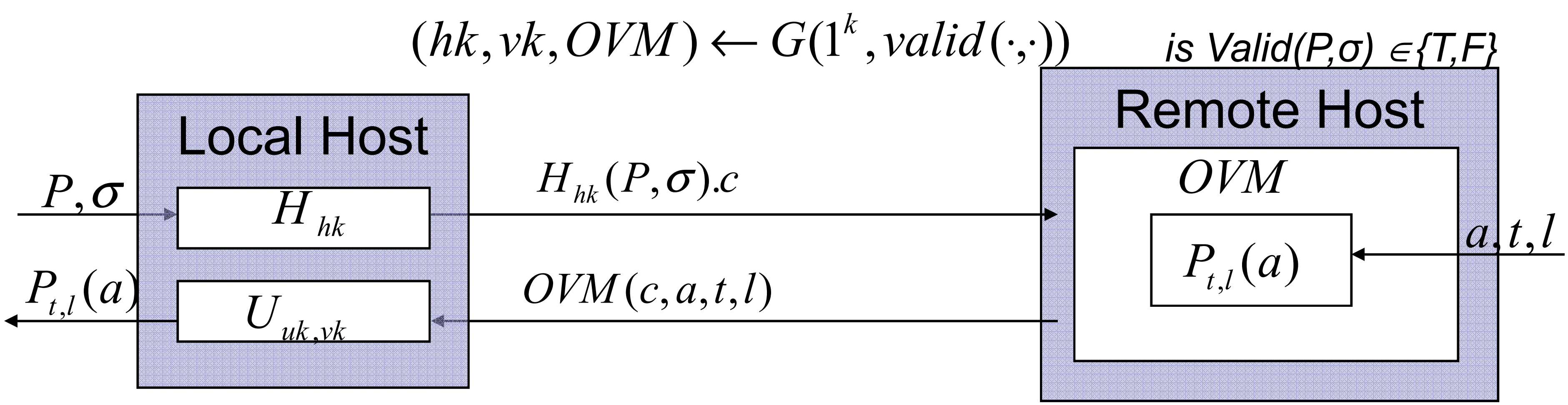


WBRPE: Separate Receiver Host



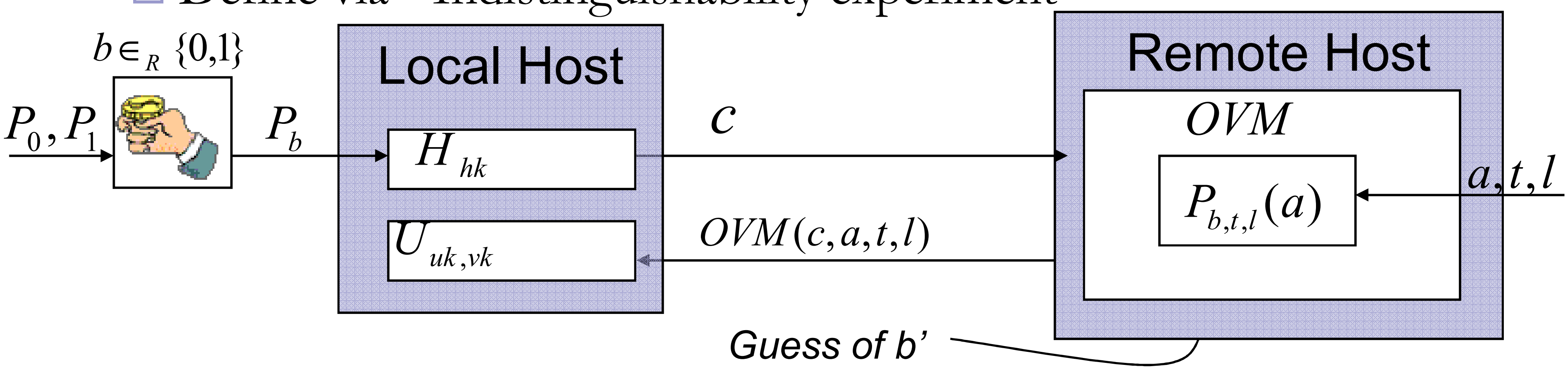
Remote Input Privacy Specification

- Protect confidentiality of inputs on the remote host
  - Expose only the output, i.e. $P_{t,l}(a)$
  - And only for a valid program P, as defined by $valid(P, \sigma)$
  - σ is a validation `hint` received along with P
    - E.g. signature on P by some program certification authority



Program Privacy (IND) Specification

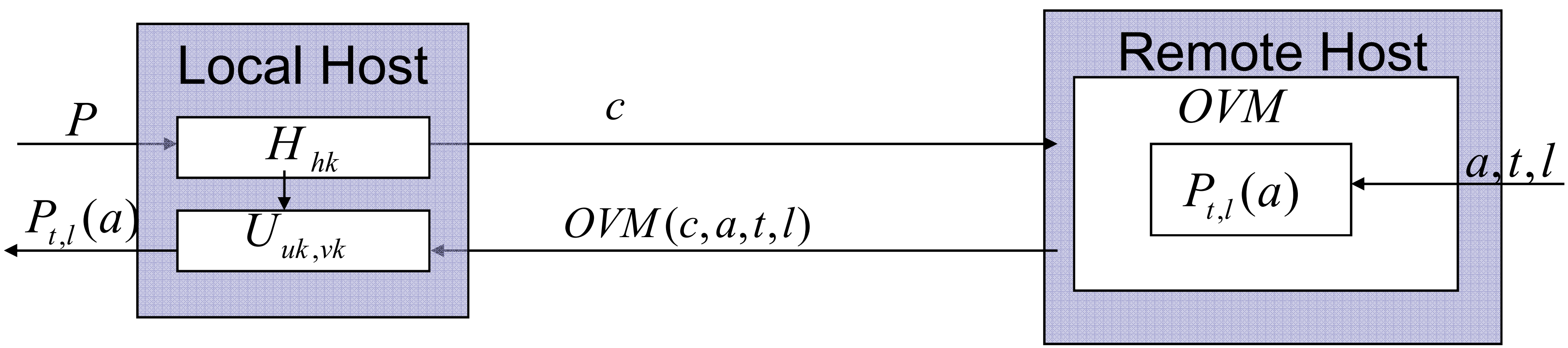
- Hide the program P input to local host
  - From malicious remote host
  - For example, hide key or data inside P
  - Define via `Indistinguishability experiment'



- Related problem: hide program from receiving host
  - Semantic-security definition

WBRPE Unforgeability Specification

- Protect the local host from malicious remote host
- Detect output forgery
  - i.e. output which is not $P_{t,l}(a)$ for any a



Talk Outline

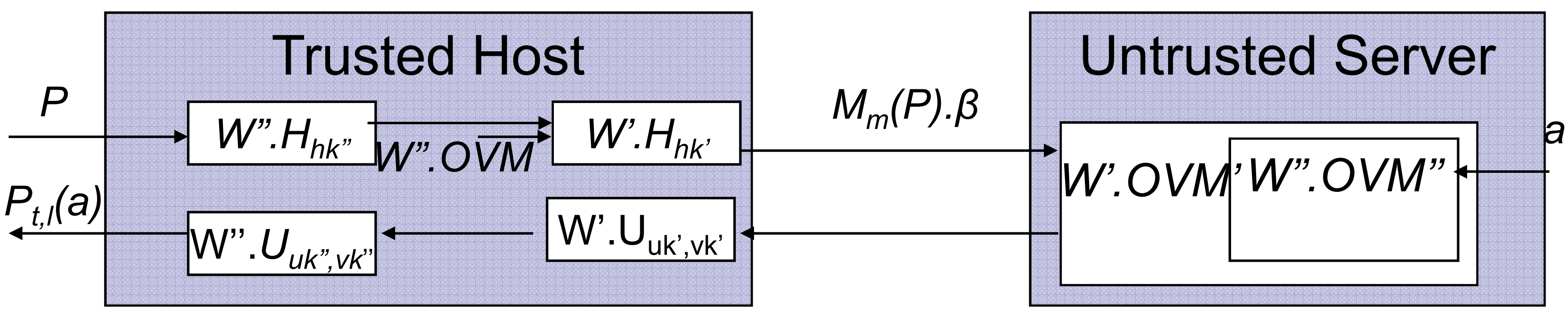
- White Box Security vs. Black Box Security
- White-Box Remote Program Execution (WBRPE)
  - Security Specifications
- WBRPE Robust Combiner
- Universal White Box RPE
- Further research and conclusions

White Box RPE Robust Combiner

- Given *two* candidate White-Box RPEs W', W''
- Can we *combine* them into one White-Box RPE
  - *W* ← *W'* • *W''*
  - s.t. *W* is a secure white box RPE provided one of *W'*, *W''* is secure
  - A robust combiner [H05]

White Box RPE Robust Combiner

- Given *two* candidate White-Box RPEs *W'*, *W''*
- Idea: run *W''* under *W'*!



White Box RPE Robust Combiner

Generation $\mathcal{G}(1^k; r_{\mathcal{G}'} || r_{\mathcal{G}''})$

$\langle hk', vk', OVM' \rangle \leftarrow \mathcal{G}'(1^k; r_{\mathcal{G}'})$

$\langle hk'', vk'', OVM'' \rangle \leftarrow \mathcal{G}''(1^k; r_{\mathcal{G}''})$

$hk = \langle hk', hk'', OVM' \rangle$

$vk = \langle vk', vk'' \rangle$

$OVM = OVM''$

$return\ \langle hk, vk, OVM \rangle$

Hardening $\mathcal{H}_{\langle hk',\ hk'' \rangle}(P; r_{\mathcal{H}'} || r_{\mathcal{H}''})$

$(c', uk') \leftarrow \mathcal{H}'_{hk'}(P; r_{\mathcal{H}'})$

```
Program P' {
    read a, t, l of the input tape
    return [OVM'(||c'||, a, t, l)];
}
```

$(c'', uk'') \leftarrow \mathcal{H}''_{hk''}(P'; r_{\mathcal{H}''})$

$uk = \langle uk', uk'' \rangle$

$return\ \langle c'', uk \rangle$

Unhardening $\mathcal{U}_{\langle (vk', uk'),\ (vk'', uk'') \rangle}(\omega) = \mathcal{U}'_{\langle vk', uk' \rangle}(\mathcal{U}''_{\langle vk'', uk'' \rangle}(\omega))$

WBRPE Combiner, Theorem

- Theorem $W \leftarrow W' \bullet W''$ is Robust for all WBRPE security specifications

- Note: significant, since we do not yet have WB solutions whose security is sufficiently established

Talk Outline

- White Box Security vs. Black Box Security
- White-Box Remote Program Execution (WBRPE)
  - Security Specifications
- WBRPE Robust Combiner
- Universal White Box RPE
- Further research and conclusions

Universal White Box RPE

- WBRPE for every program?
- We show (keyed) universal program UP_k s.t. given WBRPE for UP_k , we construct WBRPE for any program
  - UP_k is simple, efficient
- Idea: UP_k emulates the black-box solutions…

Universal program UP_k

$$\text{String } \mathtt{UP}_d = {''}\mathtt{universal\ program\ UP}_d\mathtt{(a')}\ \{$$

```
parse a' into (a, t, l, cp)
P ← D_d(cp)
y ← P_{t,l}(a)
return ⟨y, P, t⟩
}"
```

Universal WBRPE Construction

- Generation procedure

$$
\begin{array}{l}
\textbf{Program } \mathcal{G}'(1^k)\ \{ \\
\quad (hk, vk, \mathrm{OVM}_{hk}) \stackrel{R}{\leftarrow} \mathcal{G}(1^k) \\
\quad \text{String OVM'} = \text{''}\texttt{program OVM'(c,a,t,l)}\ \{ \\
\qquad \texttt{parse c} \text{ into } (\texttt{c}_{\texttt{UP}}, \texttt{c}_{\texttt{P}}) \\
\qquad \texttt{a'} \leftarrow (\texttt{a}||\texttt{t}||\texttt{l}||\texttt{c}_{\texttt{P}}) \\
\qquad \texttt{t'=t+3} \\
\qquad \texttt{l'=l+|P|+|t|} \\
\qquad \omega \leftarrow \texttt{OVM}_{\texttt{hk}}(\texttt{c}_{\texttt{UP}}, \texttt{a'}, \texttt{t'}, \texttt{l'}) \\
\qquad \texttt{return}\ \omega \\
\qquad \}'' \\
\quad \text{return } \langle hk, vk, \text{OVM'} \rangle \\
\}
\end{array}
$$

Universal WBRPE Construction

- Hardening procedure

$$
\begin{array}{l}
\textbf{Program } \mathcal{H}'(hk, P)\ \{ \\
\quad (e, d) \xleftarrow{R} \mathcal{G}_E(1^k) \\
\quad c_P \leftarrow \mathcal{E}_e(P) \\
\quad \text{Generation of UP}_d \\
\quad (c_{UP}, uk) \leftarrow \mathcal{H}_{hk}(\text{UP}_d) \\
\quad c \leftarrow (c_{UP} || c_P) \\
\quad \text{return } (c, uk) \\
\}
\end{array}
$$

- Unhardening procedure

$$
\begin{array}{l}
\textbf{Program } \mathcal{U}'(uk, vk, \omega)\ \{ \\
\quad \text{return } \mathcal{U}_{uk,vk}(\omega) \\
\}
\end{array}
$$

Talk Outline

- White Box Security vs. Black Box Security
- White-Box Remote Program Execution (WBRPE)
  - Security Specifications
- WBRPE Robust Combiner
- Universal White Box RPE
- Applications, further research and conclusions

White Box RPE Applications

- White Box RPE can be employed to address the needs of a variety of applications, e.g. :
  - Database privacy
  - Marketplace for mobile code (agents)
  - Grid computing (on demand)
  - and more…
- Other applications may require state and/or local output:
  - E-wallet, DRM,…

WBRPE Applications: DB Privacy

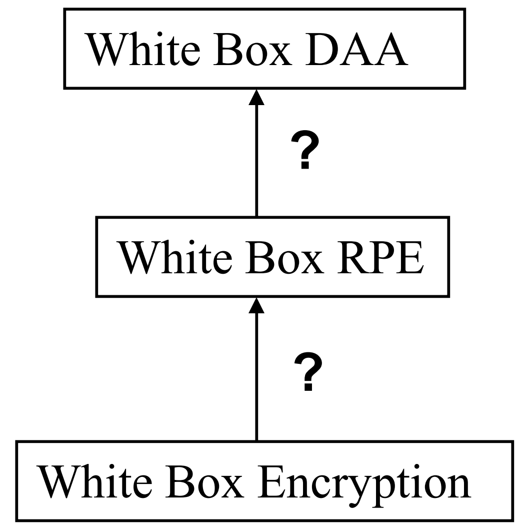
- Queries on private data base (cf. PIR)
- Legitimate query definition, privacy of the database
- How? Using certificate of valid queries
- Local host cannot learn anything about the database except result of computation

WBRPE Applications: Marketplace

- Remote host is `marketplace'
- Program (agent, mobile code) sent to remote host
  - E.g., report price changes to local host (trading user)
- Originator (local host) wants to maintain program privacy
  - E.g. to hide interests, policy, thresholds
- Marketplace/vendor/broker (remote host) may want to learn program

Future work: White Box Reductions

- Reduce (complex) tasks to WBRPE
  - E.g., for the Direct Anonymous Attestation (DAA) protocol
- Reduce WBRPE to more basic tasks
  - WB encryption ?
  - Weaker/weakest WB tasks (cf. OWF) ?
- Better `WB building blocks` (cf. to WBRPE)
  - Add state, local output



Future work: Implement WBRPE for UP_k

- From scratch (`black magic`)
- Use white box cryptography techniques
  - WB-AES, WB-DES (secure?)
- Encrypted computation
  - Several relevant results - CEF (computing with encrypted functions)
  - Efficiency concerns
- Use obfuscators

Future work: secure generation process

- Current process must be trusted by all
- Better: generation protocol involving parties
  - Each party ensures others can't cheat it
  - Cf. proactive / distributed cryptography solutions

Conclusions

- WBRPE: alternative model for SW ‘hardening’
- Presented Robust Combiner for WBRPE
  - Secure if at least one of the candidates is secure
- Presented universal program UP_k
  - WBRPE for UP_k → WBRPE for <u>all</u> programs
- Many further directions for research
- Questions?
- Thank you