Provably-secure WBRPE schemes

Amir Herzberg

Haya Shulman

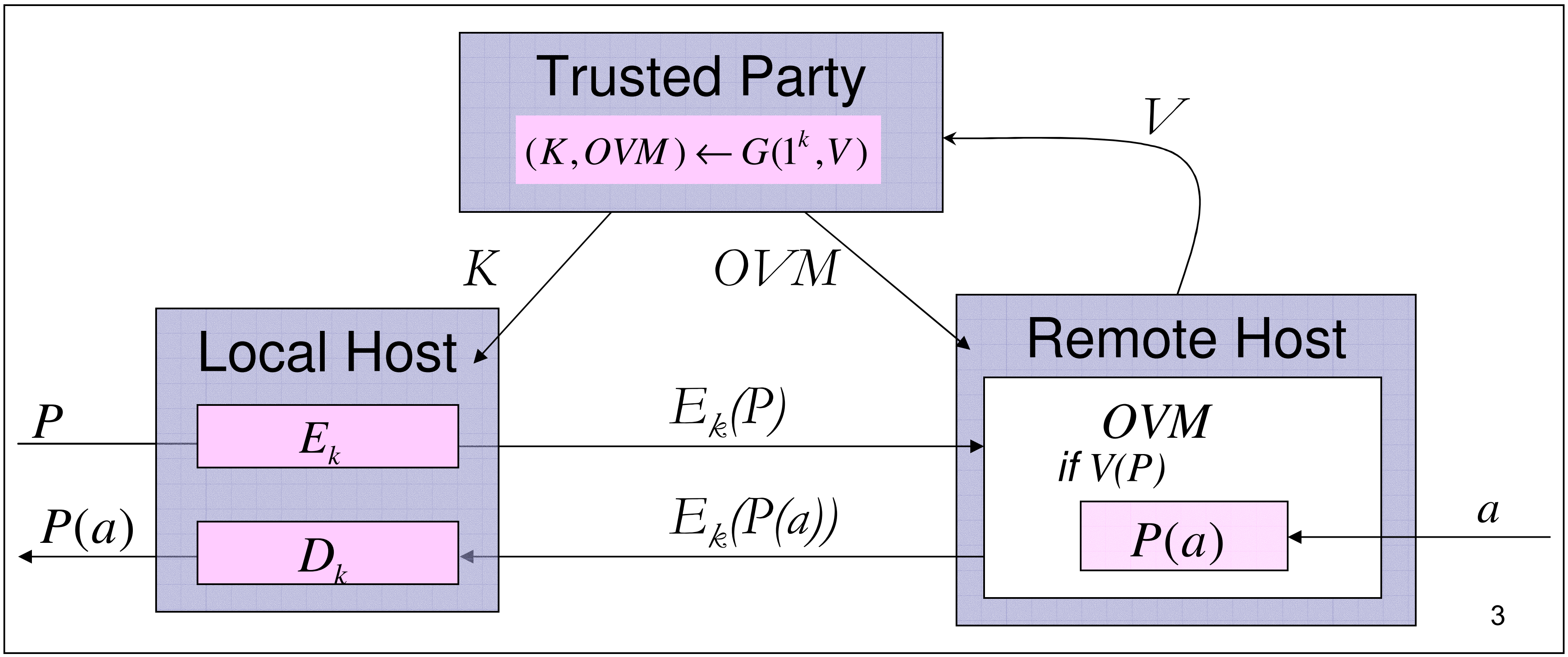
Bar Ilan university

Provably-secure WBRPE schemes: **Outline**

- Reminder:
  - WBRPE
  - Why provably-secure solutions?
- Related works (provable secure)
- Provably-secure WBRPE schemes
  - Based on Secure Function Evaluation: garbled circuits and Oblivious Transfer
  - Based on encrypted computation: homomorphic encryption ($E(x \lor y)=OR(E(x),E(y))$)
- Conclusions

WBRPE: Components (Algorithms)

- Generator G : run by Trusted Party
  - Generates key k (for local host)
  - And Obfuscated Virtual Machine OVM (for remote host)
- `Encryption' (of program sent by local host)
- `Decryption' (of result sent by remote host)



WBRPE: Goals and Results

- Reach comparable situation to cryptography:
- Provably secure WBRPE schemes
  - May not be practical (cf. [GM84, OTP])
- Practical, efficient, cryptanalysis-proven WBRPE schemes
  - Secure by evidence of failed cryptanalysis, safety margins
- Results
  - Theoretical feasibility results (provably secure schemes)
  - Robust combiner: given two candidate WBRPE schemes, build one that is secure – if one of the two candidate schemes is secure
    - Allows safety-margins in design

Why Provably-Secure Solutions?

- Most white-box security work is heuristic
- This talk focuses on provably-secure WBRPE
- Why is it interesting?
  - Theoretical interest: is it possible?
    - Doubts raised after presenting WBRPE defs
  - `Feasibility proof`
    - Encourages search for practical (heuristic?) schemes
  - Ideas for design of practical WBRPE schemes?

Related Works

- Secure multiparty/two-party computations [Yao, GMW,…]
- Secure function evaluation based on Yao's garbled circuits [Cachin, Camenisch, Kilian, Muller]
- Encrypted computing, based on homomorphic encryption [Sander, Young, Yung]
- More [see paper]

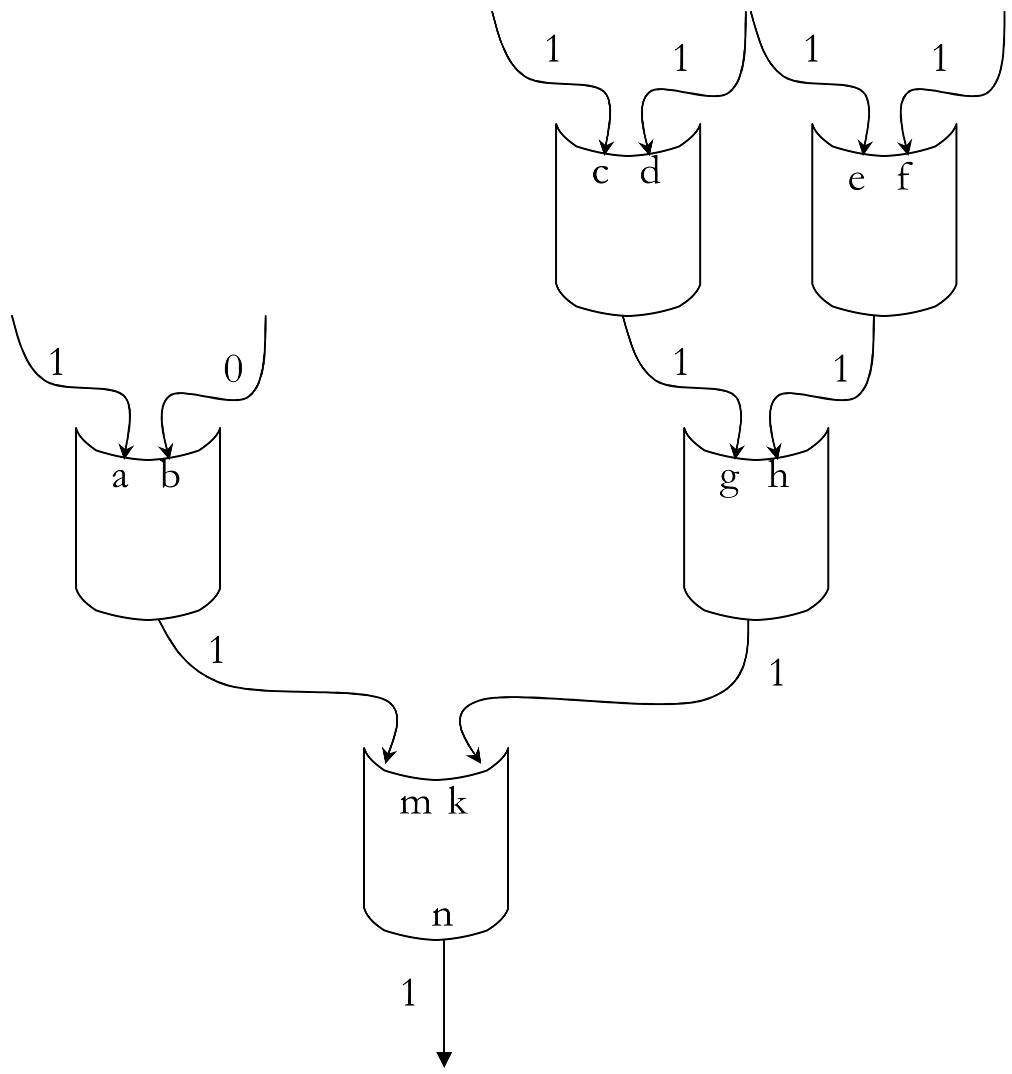
Provably-secure WBRPE schemes: **Outline**

- Reminder:
  - WBRPE
  - Why provably-secure solutions?
- Related works (provable secure)
- **Provably-secure WBRPE schemes**
  - **Based on Secure Function Evaluation: garbled circuits and Oblivious Transfer**
    - **Tools**
    - Design
  - Based on encrypted computation: homomorphic encryption $(E(x \vee y)=OR(E(x),E(y))$
- Conclusions

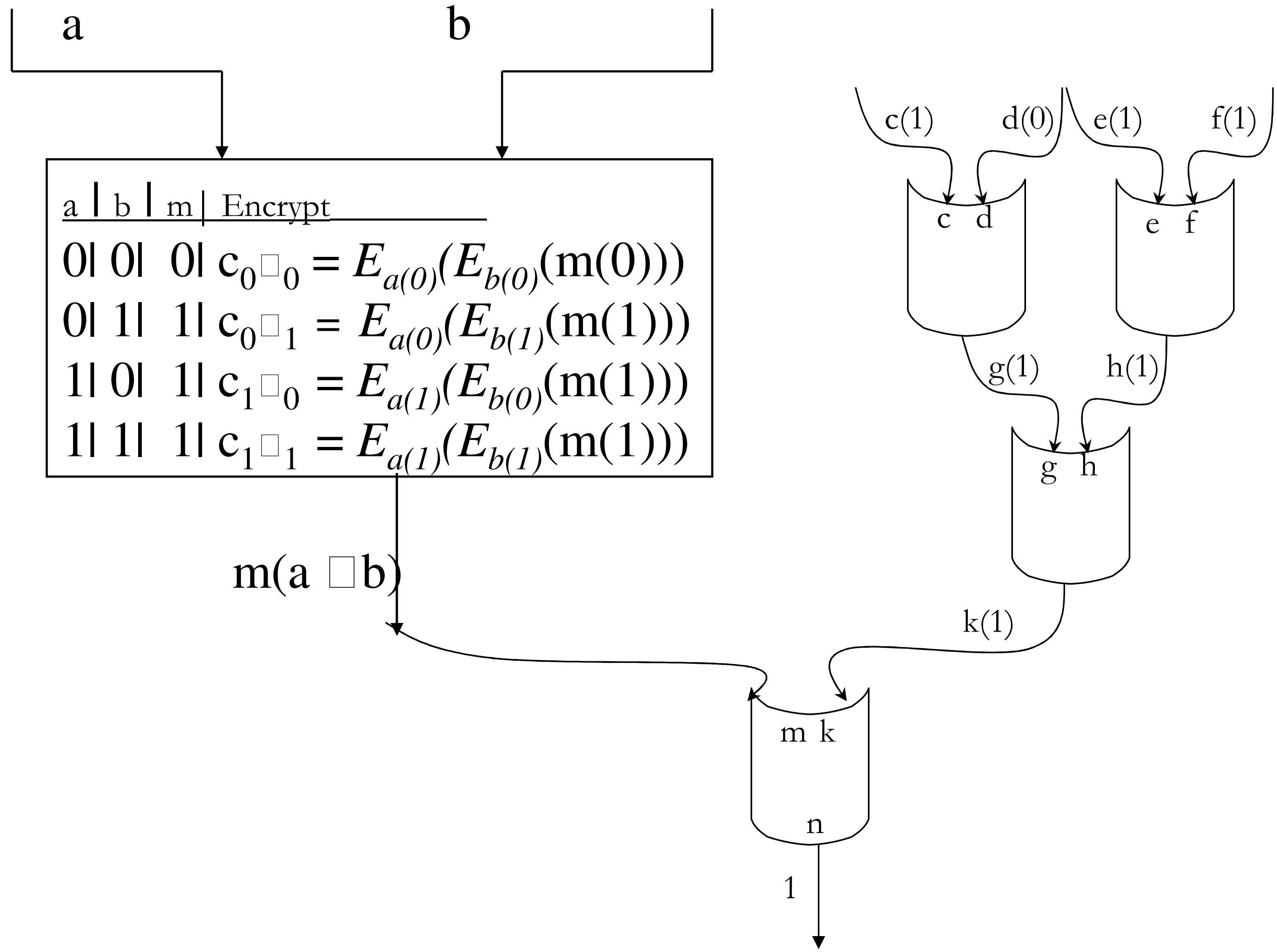
Cryptographic Tools: Garbled Circuit

- Yao's secure circuit evaluation protocol allows two parties with secret inputs each to evaluate poly-size Boolean circuit
- Represent $f(\cdot,\cdot)$ as a Boolean circuit
- "garbling" of the circuit: replace every gate in the circuit with encrypted version of the gates
  - ∀ wire, assign random strings representing 0/1
  - ∀ gate, construct a "secure" garbled truth table
- At every step, the party evaluating the circuit, computes some function of its secret input and the encrypted value received from other party

Boolean OR Circuit:

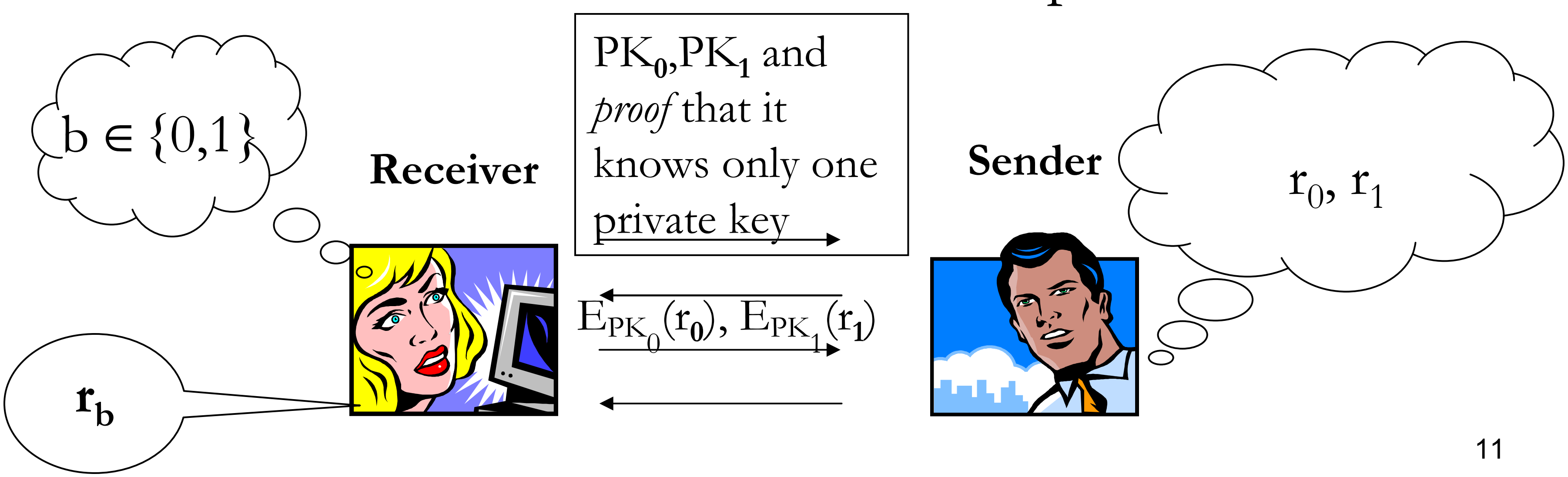


Boolean OR Circuit:



Cryptographic Tools: Oblivious Transfer

- Can be based on most public-key systems
- The sender has two inputs, and the receiver wants to learn one of them, at the end of the protocol:
  - the receiver learns this input and nothing else
  - the sender should not learn which input this was

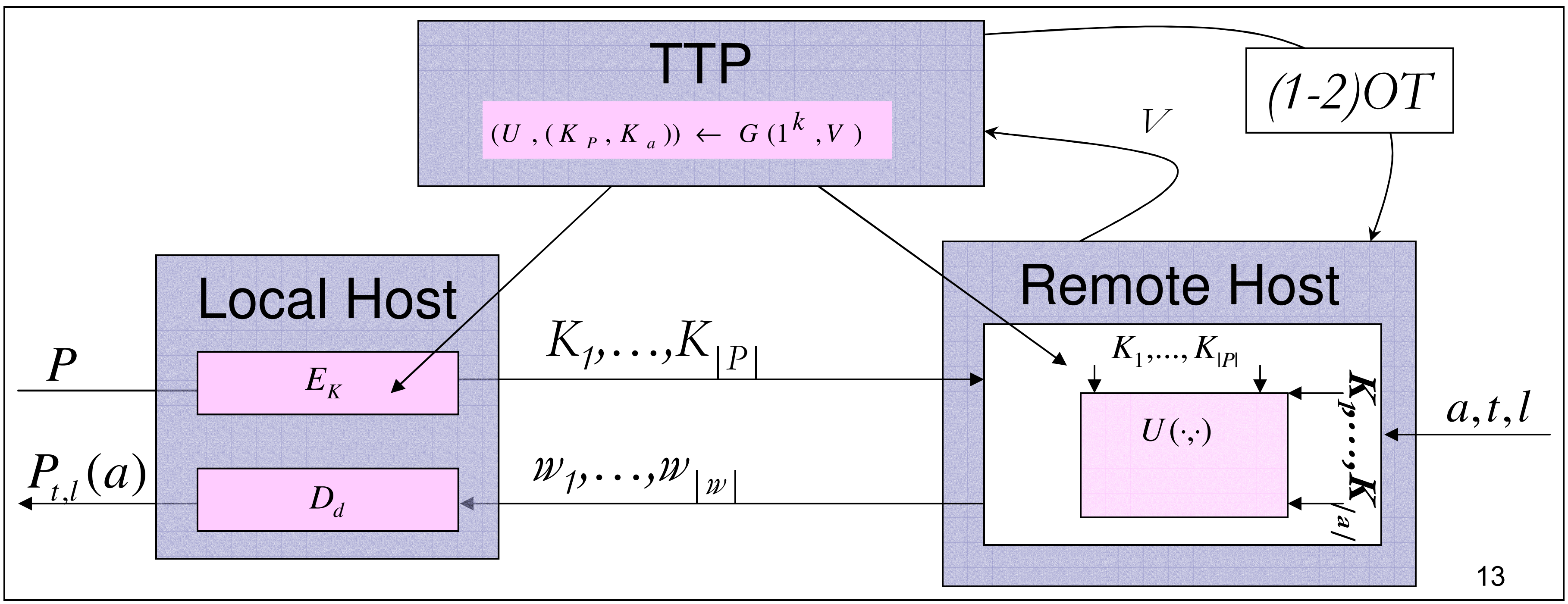


Provably-secure WBRPE schemes: **Outline**

- Reminder:
  - WBRPE
  - Why provably-secure solutions?
- Related works (provable secure)
- **Provably-secure WBRPE schemes**
  - **Based on Secure Function Evaluation: garbled circuits and Oblivious Transfer**
    - Tools
    - **Design**
  - Based on encrypted computation: homomorphic encryption ($E(x \vee y) = OR(E(x), E(y))$)
- Conclusions

WBRPE Based on Garbled Circuit

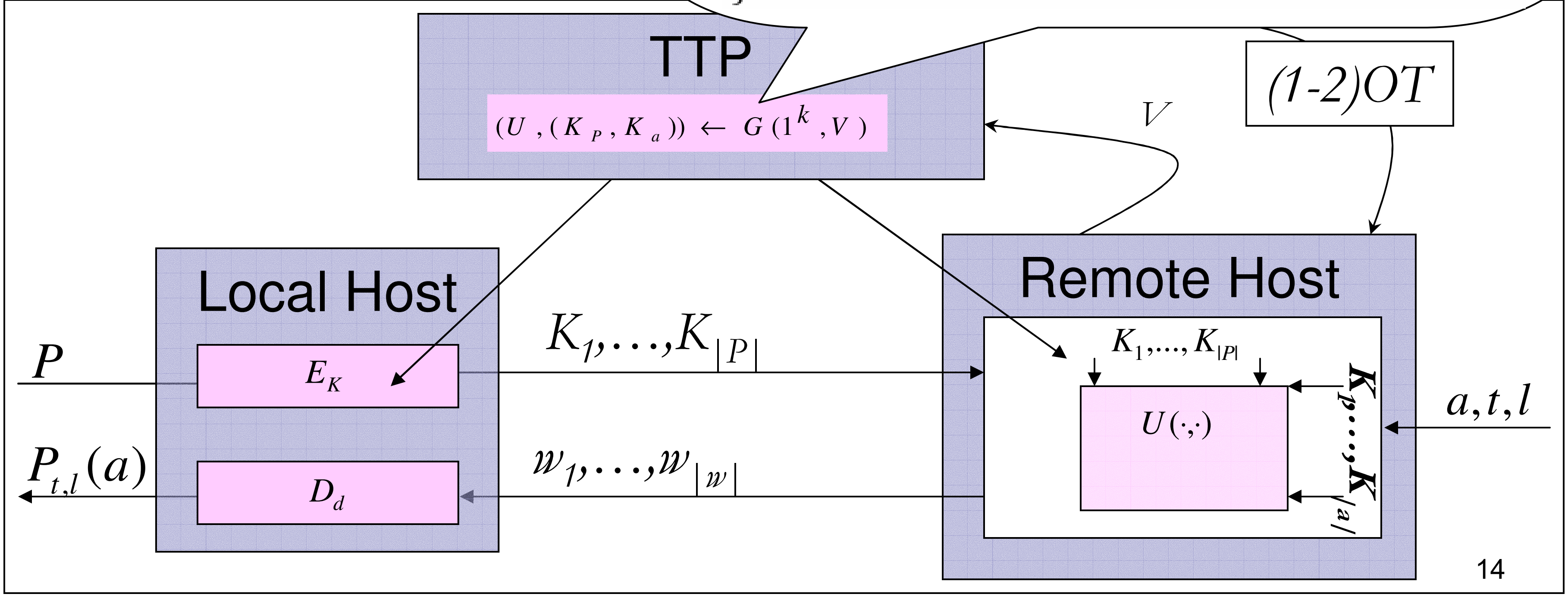
- To achieve privacy of both parties, use third party T
  - T generates garbled circuit, and random keys
  - Local receives output decryption tables, and random strings for P
  - Remote receives garbled tables



WBRPE Based on Garbled Circuit

- To achieve privacy of b
  - T generates garbled circ
  - Local receives output d
  - Remote receives garble

$\mathcal{G}(1^k, V)$ {
$\quad (K_P, K_a) \leftarrow G_E(1^k)$
$\quad \mathcal{U} \leftarrow createU(V)$
$\quad K_P = ((K_{1,0}, K_{1,1}), ..., (K_{|P|,0}, K_{|P|,1}))$
$\quad K_a = ((K_{1,0}, K_{1,1}), ..., (K_{|a|,0}, K_{|a|,1}))$
$\quad \mathcal{U} \leftarrow Garble(U)$
return $(\mathcal{U}, K_p, K_a)$
}

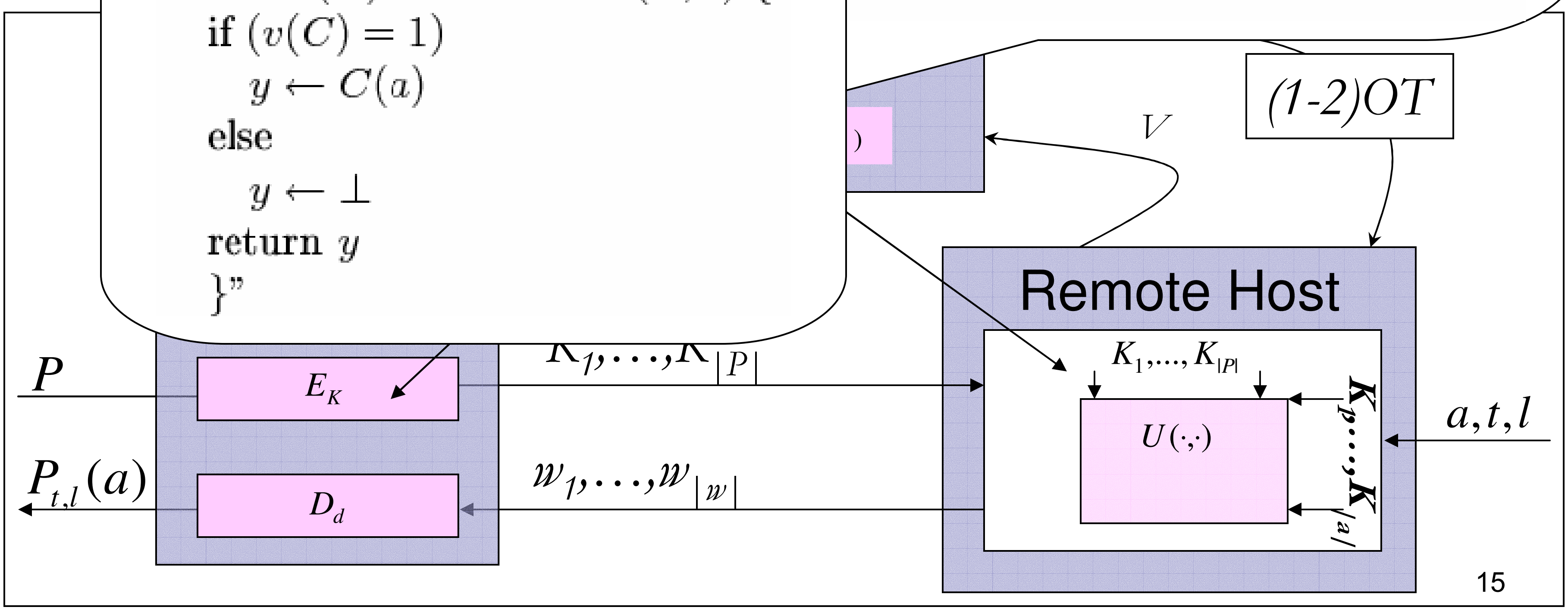


WBRPE Based on Garbled Circuit

- To achieve privacy of b
  - T generates garbled circ
  - Local receives output d
  - Remote receives garb

$\mathcal{G}(1^k, V)$ {
$(K_P, K_a) \leftarrow G_E(1^k)$
$\mathcal{U} \leftarrow createU(V)$
$K_P = ((K_{1,0}, K_{1,1}), ..., (K_{|P|,0}, K_{|P|,1}))$
$K_a = ((K_{1,0}, K_{1,1}), ..., (K_{|a|,0}, K_{|a|,1}))$
$\mathcal{U} \leftarrow Garble(U)$
return $(\mathcal{U}, K_p, K_a)$

$createU(V) = return\ "U(C, a)$ {
if $(v(C) = 1)$
$y \leftarrow C(a)$
else
$y \leftarrow \perp$
return y
}"

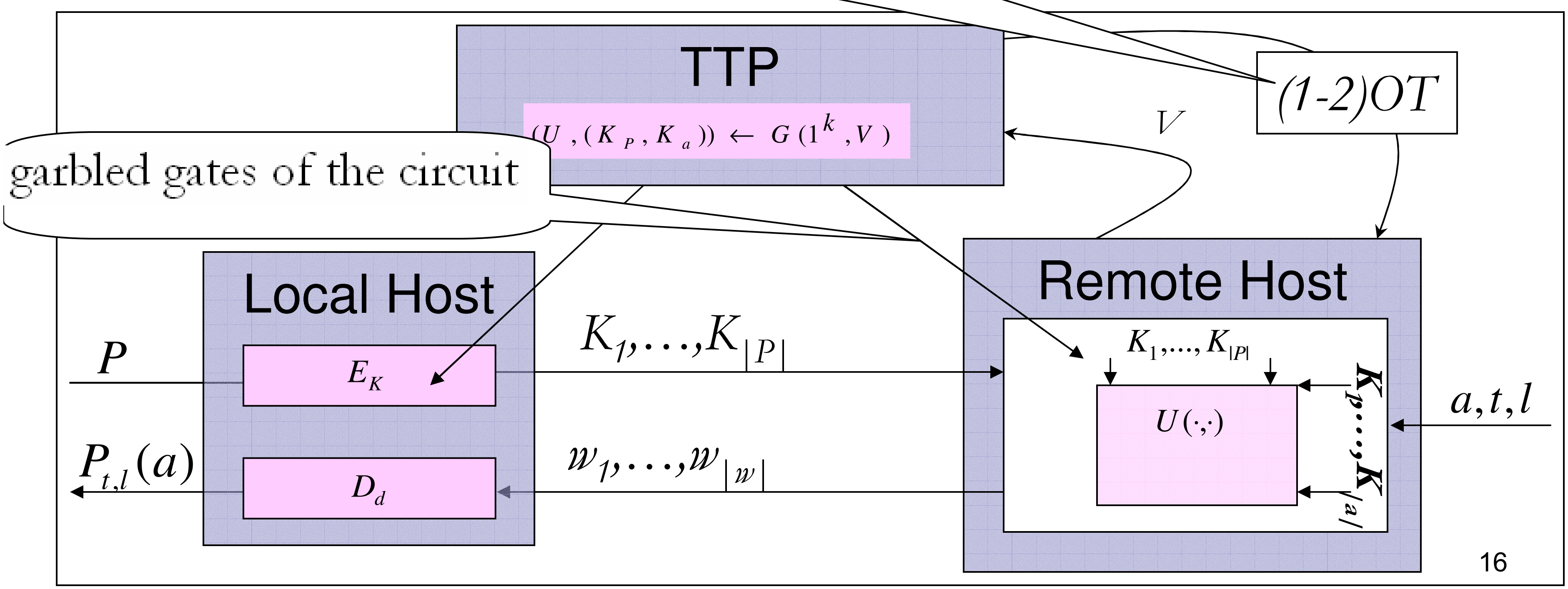


WBRPE Based on Garbled Circuit

- To achieve privacy of both parties, use third party T
  - T generates garbled circuit, and random keys
  - Local … d random strings for P
  - Remo…

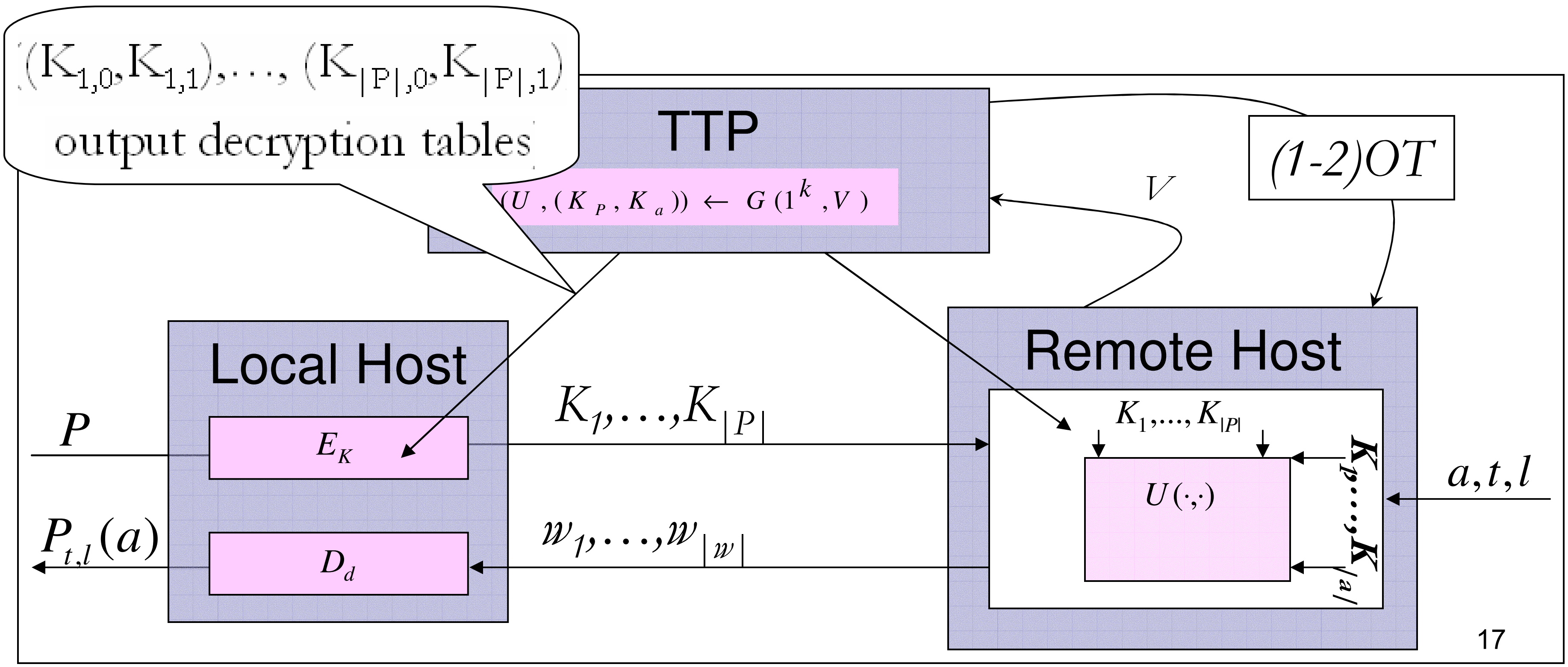
Transfer $(x_i,(K_{i,0},K_{i,1}))$:

- Interactive (1-2)OT protocol between local and remote for each wire of the circuit



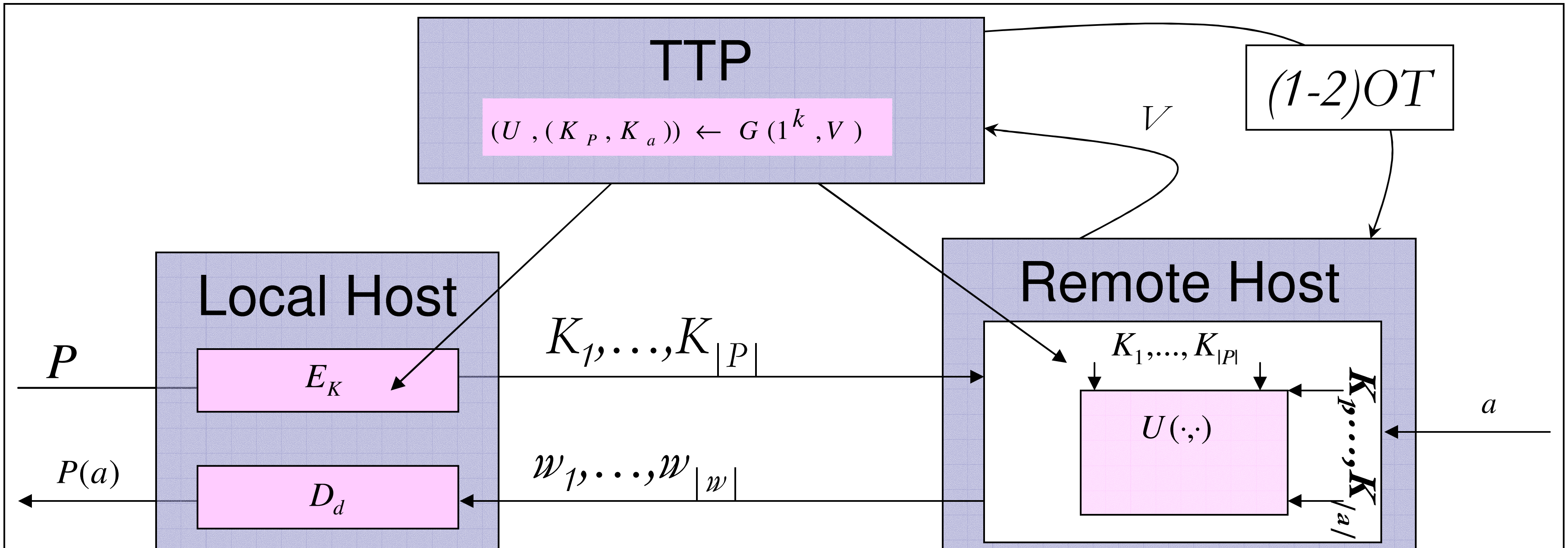
WBRPE Based on Garbled Circuit

- To achieve privacy of both parties, use third party T
  - T generates garbled circuit, and random keys
  - Local receives output decryption tables, and random strings for P
  - Remote receives garbled tables



WBRPE Based on Garbled Circuit: Properties

- Privacy of local and remote inputs
- Unforgeability: since remote only gets random strings corresponding to correct inputs, can't compute incorrect outputs
- Single program *P*; remote commits to *a* at generation



If Third Party is not Trusted…

- Cut-and-Choose:
  - Remote parses C into m garbled circuits, and sends them to Alice. Alice also parses C.
  - Alice chooses one circuit for evaluation – ***C***
  - Bob exposes secrets of all garbled circuits except ***C***
  - Alice verifies all exposed garbled circuits
  - Catches cheating with probability 1-1/m
- Bob sends his inputs for ***C*** (Alice can't interpret them because they are garbled)

Non-Interactive Encrypted Computation

- Computing with Encrypted Data (CED)
  - Local host has input *x*, remote has a function *f*
  - Local sends $E(x)$ to remote
  - Remote computes $E(f(x))$ and sends to local
  - Local decrypts and learns $f(x)$ in one-round protocol

- Computing with Encrypted Functions (CEF)
  - Local has a function *f*, remote has an input *x*
  - Local encrypts *f* an generates a program $P(E(f))$
  - Remote computes $P(E(f))(x)$ and returns to local
  - Local decrypts $P(E(f))(x)$ and obtains $f(x)$

Provably-secure WBRPE schemes: **Outline**

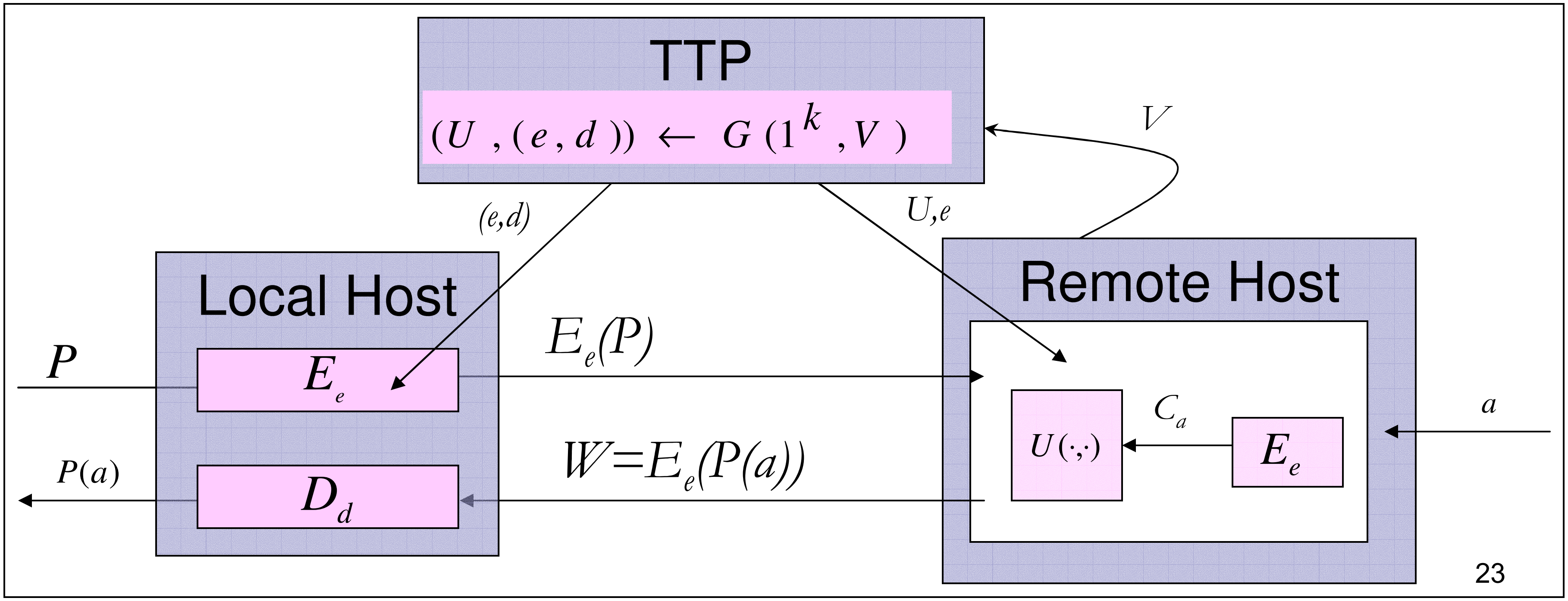
- Reminder:
  - WBRPE
  - Why provably-secure solutions?
- Related works (provable secure)
- **Provably-secure WBRPE schemes**
  - Based on Secure Function Evaluation: garbled circuits and Oblivious Transfer
  - **Based on encrypted computation: homomorphic encryption ($E(x \vee y)=OR(E(x),E(y))$)**
- Conclusions

WBRPE Based on Encrypted Computation

- For circuits with limited depth
- Technique to securely evaluate circuit
  - Using probabilistic homomorphic encryption
  - Allowing efficient computation of NOT and OR gates for encrypted values:
    - E(~a)=NOT(E(a)),
    - E(a□b)=OR(E(a),E(b))

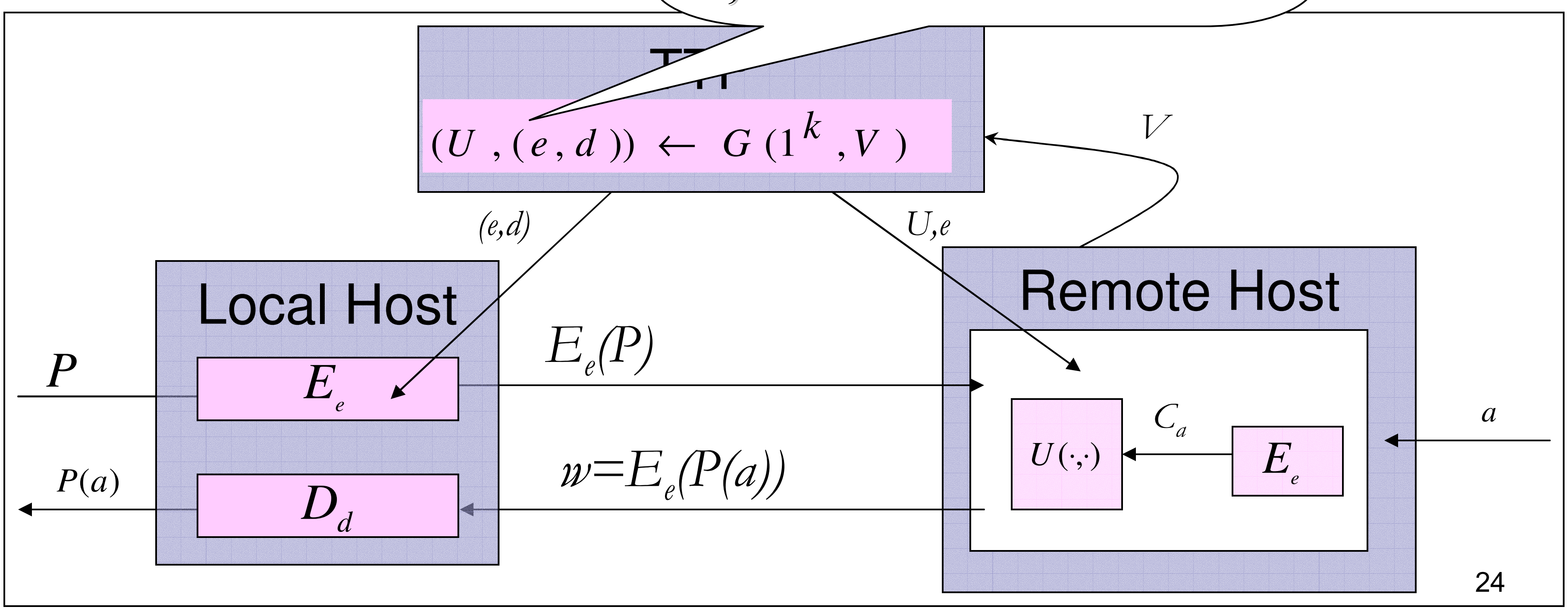
WBRPE Based on Encrypted Computation

- To achieve privacy of both parties, use third party TTP
  - TTP generates universal circuit U, and pair of keys (e,d)
    - $U(E_e(P),E_e(a))=E_e(P(a))$ if $V(P)=OK$.
  - Local receives (e,d)
  - Remote U and e



WBRPE Based on Encrypted Computation

- To achieve privacy of both parties, use third party T
  - T generates universal ci
  - Local receives (e,d)
  - Remote U and e

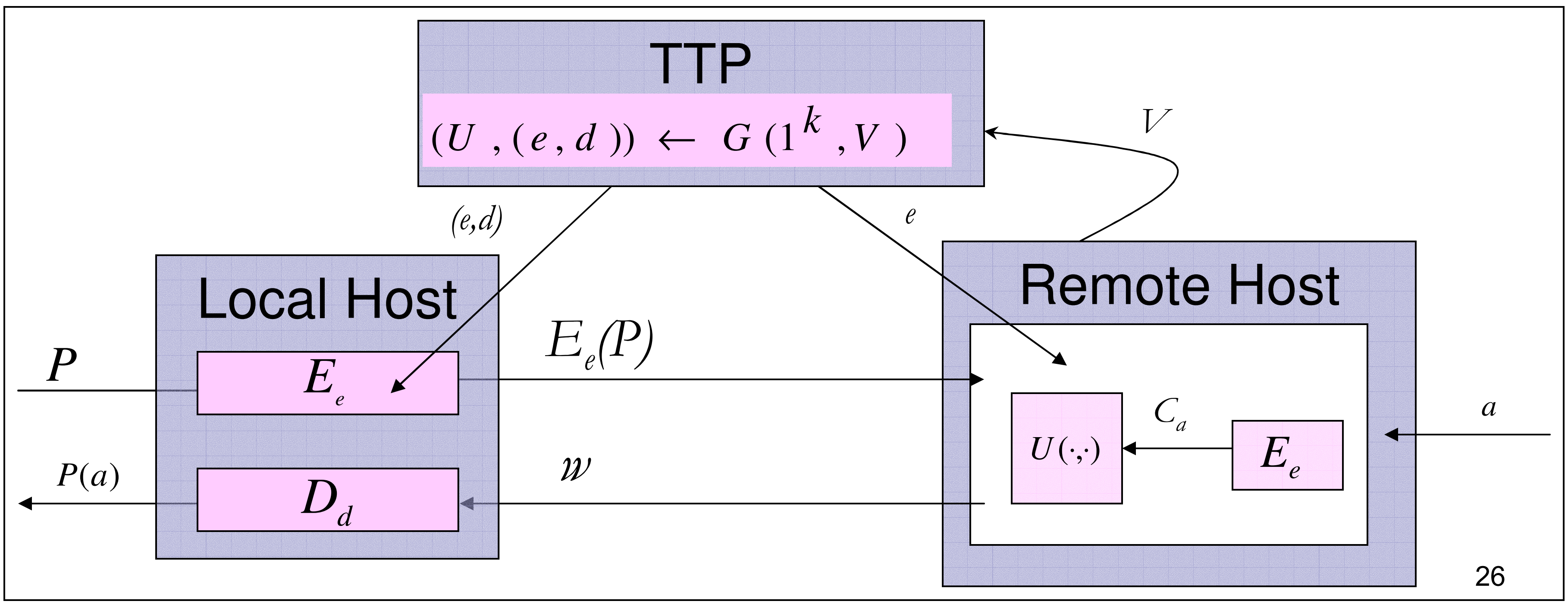


WBRPE Based on Encrypted Computation

- TTP receives predicate V from remote and generates
  - Universal circuit $U(P,a)=P(a)$ if $V(P)=1$
  - Keys (e,d)
- TTP sends
  - U and e to remote
  - (e,d) to local
- Local sends to the remote host encryptions of each bit of the program, i.e. $c_P[i]=E_e(P_i)$
  - Remote host bit-wise encrypts $c_a=E_e(a)$, evaluates $w=U(c_P, c_a)$, and returns w to local
  - Local recovers y using d

Encrypted Computation WBRPE: Properties

- Privacy of local input *P*: since outputs are encrypted
- Privacy of remote input *a*: since *U* validates *P* with *V*
- Integrity? Local host sends `dummy computations`
- Efficiency?



WBRPE Based on Encrypted Computation

- Efficiency?
  - Expansion of the sizes of outputs relative to sizes of inputs
    - Worst case $8^{\{circuit\ depth\}}$
    - For iterative applications required space and computation complexity grows exponentially with each step → only works for small circuits

Provably-Secure WBRPE: Conclusions

- Presented two provably-secure WBRPE schemes
  - Based on garbled circuits
    - For one program only, commit to *a* before generation
  - Based on homomorphic encryption
    - Only for programs encoded by polylog circuit
- Open questions
  - Better provably secure WBRPE schemes (e.g., arbitrary circuits)