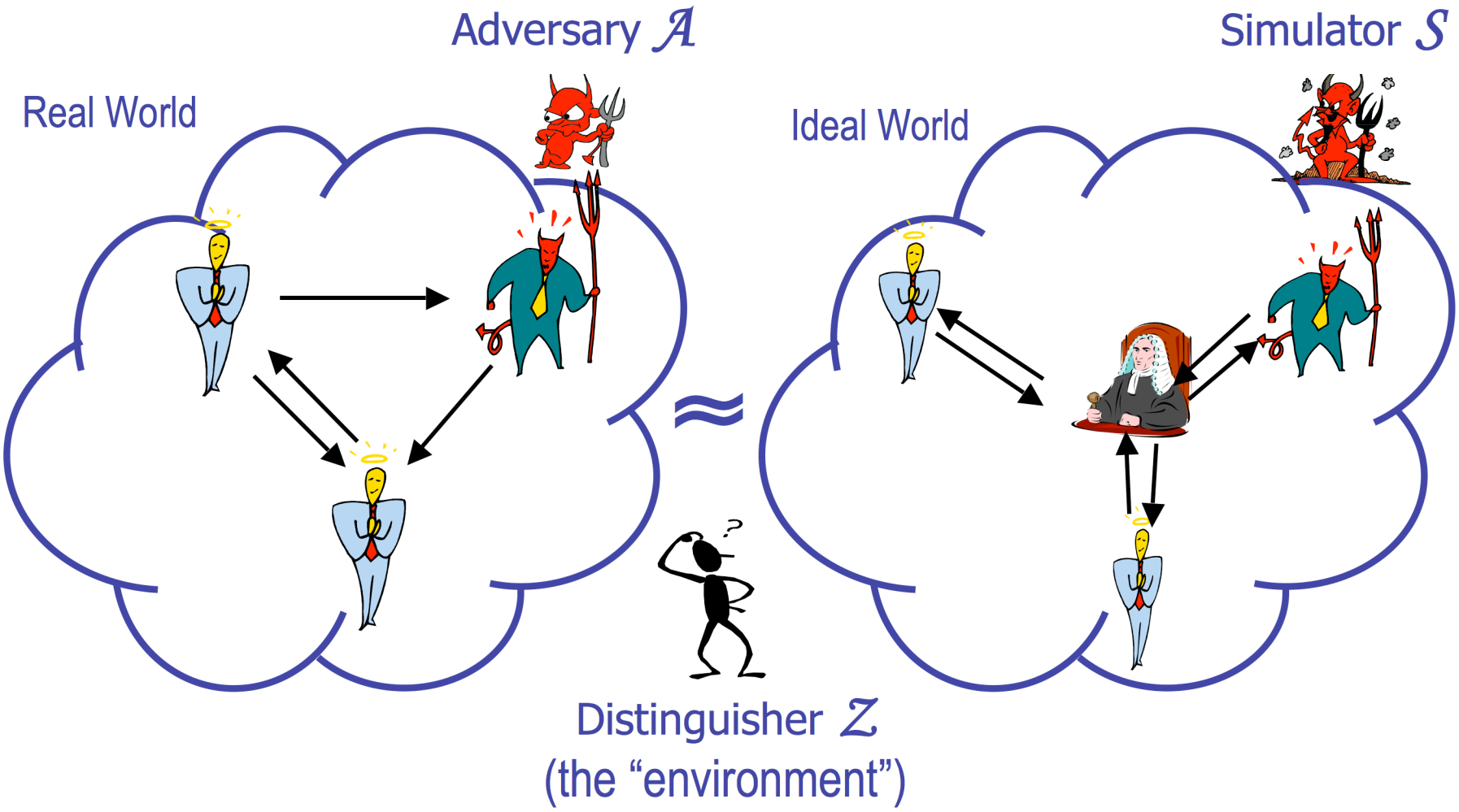
Sound and Fine-grain Specification of Ideal Functionalities

Juan Garay (*Bell Labs*)
Aggelos Kiayias (*University of Connecticut*)
Hong-Sheng Zhou (*University of Connecticut*)

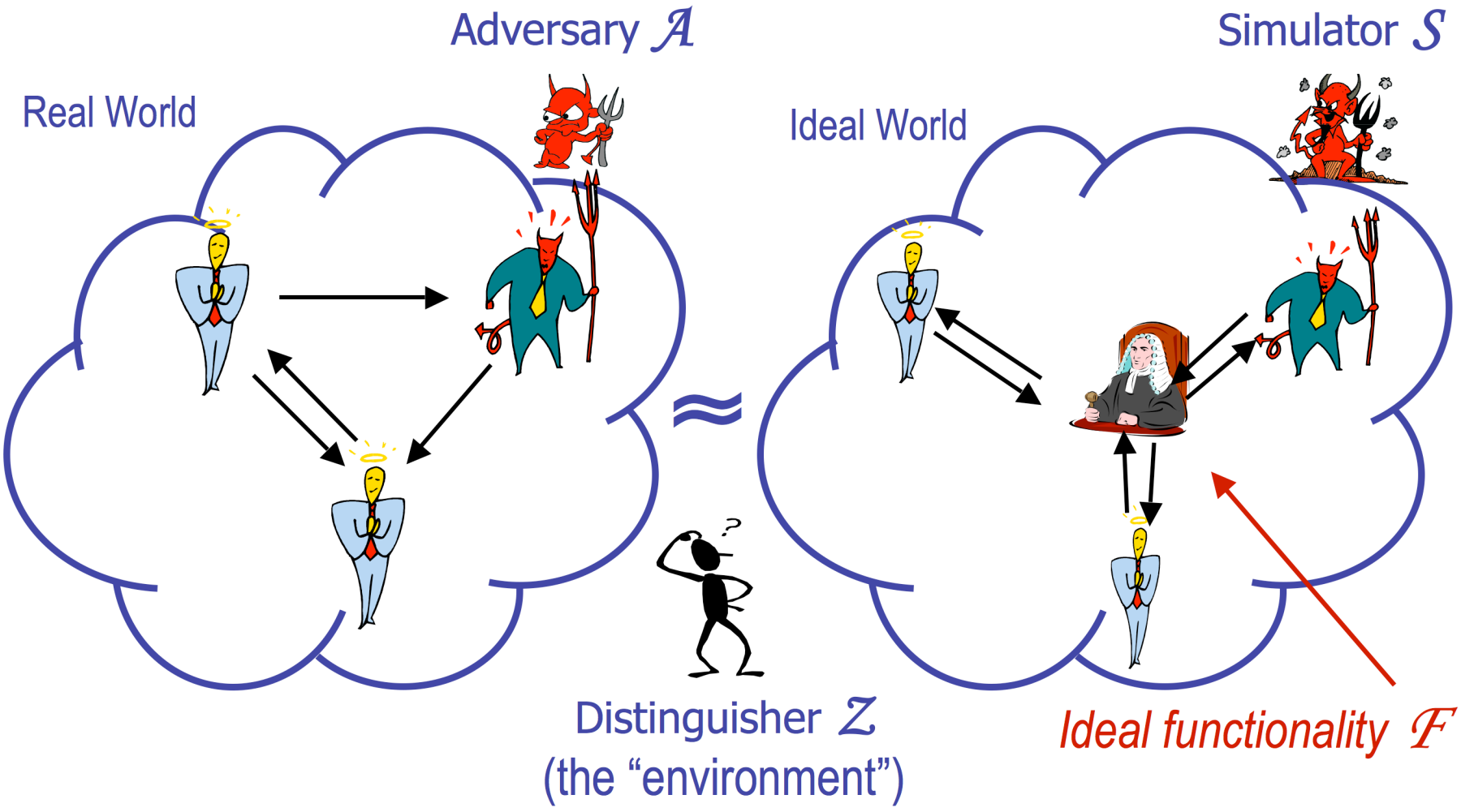
Universal Composability (UC) Framework

- Guarantees Strong Security Properties (Concurrent Composition, Non-malleability, etc.)
- Simulation-based

UC Framework



UC Framework



Ideal Functionalities

i·de·al (ī-dē'əl, ī-dēl') 

n.

1. A conception of something in its absolute perfection.
2. One that is regarded as a standard or model of perfection or excellence.
3. An ultimate object of endeavor; a goal.
4. An honorable or worthy principle or aim.

Ideal Functionalities

i·de·al (ī-dē'əl, ī-dēl') 

n.

1. A conception of something in its absolute perfection.
2. One that is regarded as a standard or model of perfection or excellence.
3. An ultimate object of endeavor; a goal.
4. An honora

?

Ideal Functionalities

i·de·al

n.

1. A c
2. One
3. An
4. An

The ideal process

Participants: Environment $\mathcal{Z}$ and ideal-process adversary $\mathcal{S}$, interacting with ideal functionality $\mathcal{F}$ and dummy parties $\tilde{P}_1, ..., \tilde{P}_n$. All participants have the security parameter k; $\mathcal{Z}$ also has input z.

1. While $\mathcal{Z}$ has not halted do:
   (a) $\mathcal{Z}$ is activated (i.e., its activation tape is set to 1). $\mathcal{Z}$'s activity remains unchanged from the real-life model (Figure 1). In particular, $\mathcal{Z}$ cannot access $\mathcal{F}$.
   (b) Once a dummy party $\tilde{P}_i$ is activated with a new value on its input tape, it copies this value to the incoming communication tape of $\mathcal{F}$, and enters the waiting state. $\mathcal{F}$ is activated next.
   (c) Once $\mathcal{F}$ is activated it follows its program until it enters either the waiting state or the halt state. In particular, $\mathcal{F}$ may write on its outgoing communication tape messages addressed to the parties and adversary. If $\mathcal{F}$ wrote a message to the adversary then the adversary is activated next. Otherwise, the party that was last activated before $\mathcal{F}$ is activated again.
   (d) Once the adversary $\mathcal{S}$ is activated, it follows its program and possibly writes new information on its output tape. In addition, $\mathcal{S}$ can perform one of the following activities.
      i. $\mathcal{S}$ may ask to see the contents of the outgoing communication tapes of the dummy parties, and the destinations of the outgoing messages generated by $\mathcal{F}$. The desired information is then written on $\mathcal{S}$'s incoming communication tape.
      ii. $\mathcal{S}$ may deliver a message m from $\mathcal{F}$ to some dummy party $\tilde{P}_i$. Here $\mathcal{S}$ does not have access to the contents of m (unless $\tilde{P}_i$ is corrupted); it only sees the destination of m.
      iii. $\mathcal{S}$ may write a message, m, on the incoming communication tape of $\mathcal{F}$. This message appears with sender $\mathcal{S}$.
      iv. $\mathcal{S}$ may corrupt a dummy party $\tilde{P}_i$. Upon corruption, $\mathcal{Z}$ and $\mathcal{F}$ are notified of the corruption event, and $\mathcal{F}$ may hand $\mathcal{S}$ some information regarding the internal state of $\tilde{P}_i$. From this point on, P_i may no longer be activated; also, $\mathcal{S}$ receives all the messages from $\mathcal{F}$ that are addressed to P_i, and may deliver to $\mathcal{F}$ messages whose sender is P_i.

   If some message was delivered to $\tilde{P}_i$ (resp., $\mathcal{F}$) in this activation then $\tilde{P}_i$ (resp., $\mathcal{F}$) is activated once $\mathcal{S}$ enters the waiting state. Otherwise, $\mathcal{Z}$ is activated next.
   (e) Once a dummy party $\tilde{P}_i$ is activated with a new incoming message from $\mathcal{F}$ it copies this message to its output tape and enters the waiting state. $\mathcal{Z}$ is activated next.
2. The global output is the first bit of the output tape of $\mathcal{Z}$.

?

Traditional Security Notions

- CPA/CCA
- CMA Security
- SK Security
- ZK/WI/Soundness
- Hiding/Binding/NM
- …..

- Well-understood.
- *On the other hand:* Definition of UC functionalities error-prone / leads to unstable definitions.

→ **Needed** : Systematic way to *translate* traditional security notions

This Work

This Work

- Language-theoretic description of functionalities

This Work

- Language-theoretic description of functionalities
- Bridge traditional defs to UC defs systematically

This Work

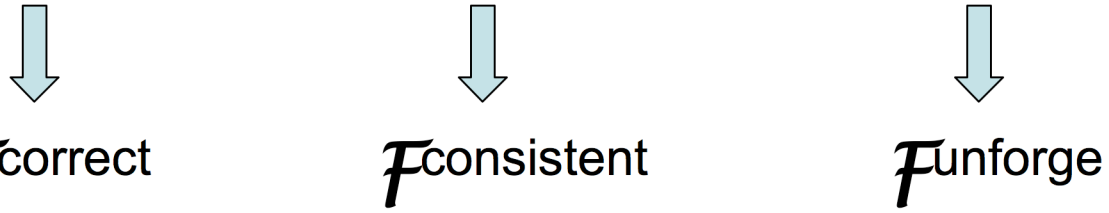
- Language-theoretic description of functionalities
- Bridge traditional defs to UC defs systematically
- Allows for
  - modular and non-modular design at the same time.
  - Identifying relations between functionalities easily.
  - “debug” existing functionalities.

Example: Secure Sig to $\mathcal{F}_{\text{SIG}}$

SS = correctness + consistency + unforgeability

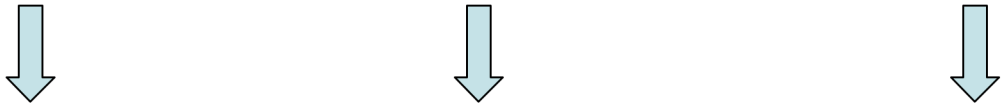
Example: Secure Sig to $\mathcal{F}_{SIG}$

SS = correctness + consistency + unforgeability



Example: Secure Sig to $\mathcal{F}_{SIG}$

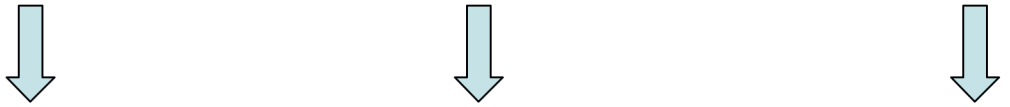
SS = correctness + consistency + unforgeability



$\mathcal{F}^{SS} = \mathcal{F}^{correct} \cap \mathcal{F}^{consistent} \cap \mathcal{F}^{unforge}$

Example: Secure Sig to $\mathcal{F}_{SIG}$

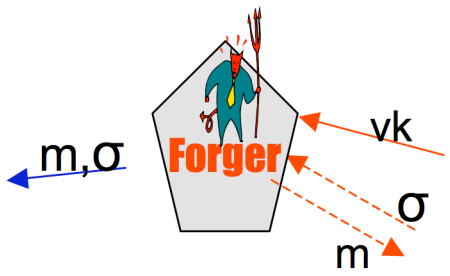
SS = correctness + consistency + unforgeability



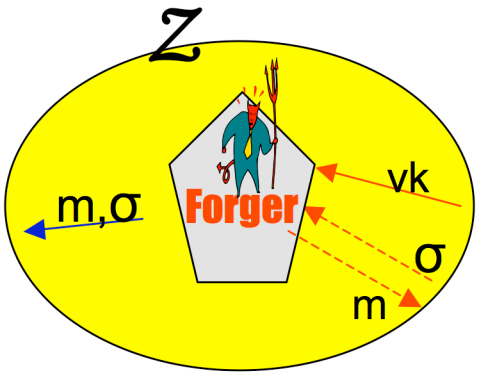
$\mathcal{F}^{SS} = \mathcal{F}^{correct} \cap \mathcal{F}^{consistent} \cap \mathcal{F}^{unforge}$

$\mathcal{F}_{SIG1}$ [c01], $\mathcal{F}_{SIG2}$ [c04] $\in \mathcal{F}^{SS}$

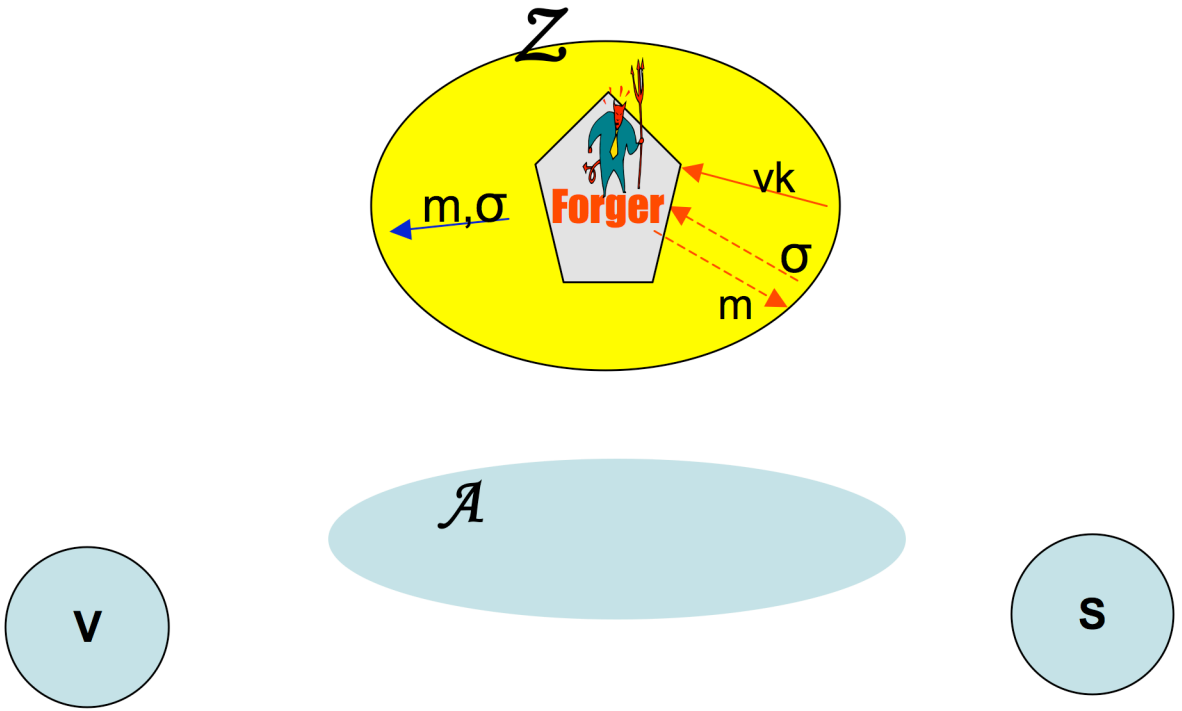
From Unforgeability to $\mathcal{F}^{\text{unforge}}$



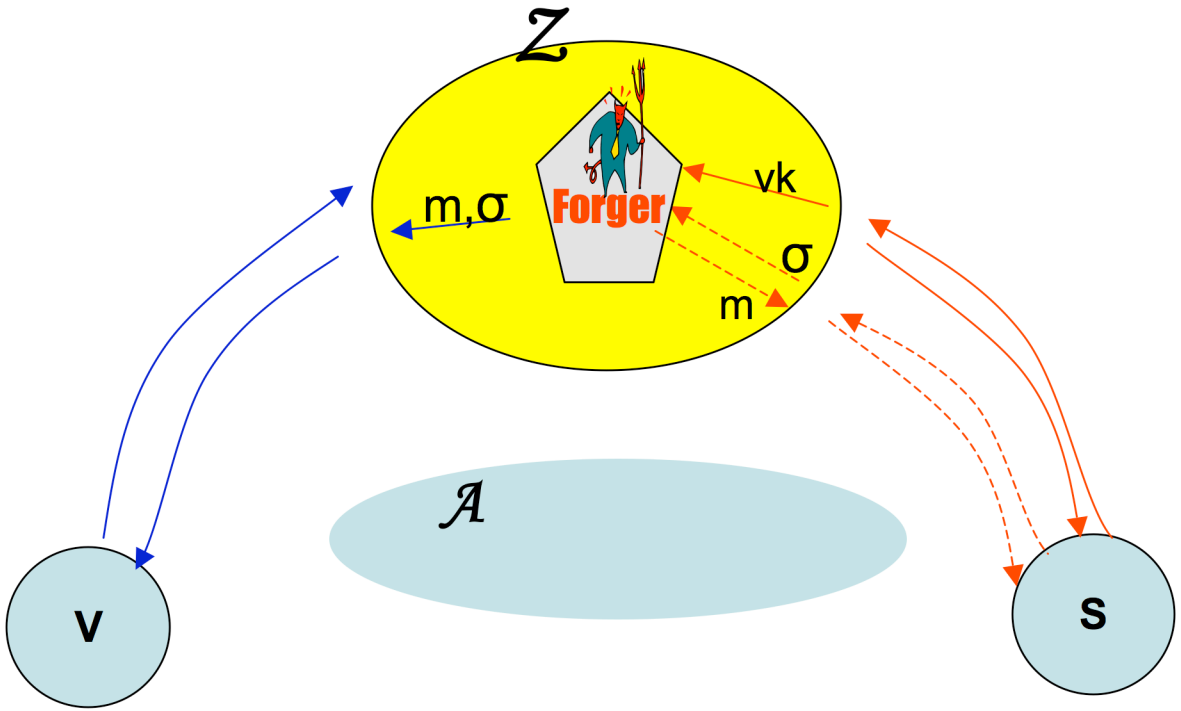
From Unforgeability to $\mathcal{F}^{\text{unforge}}$



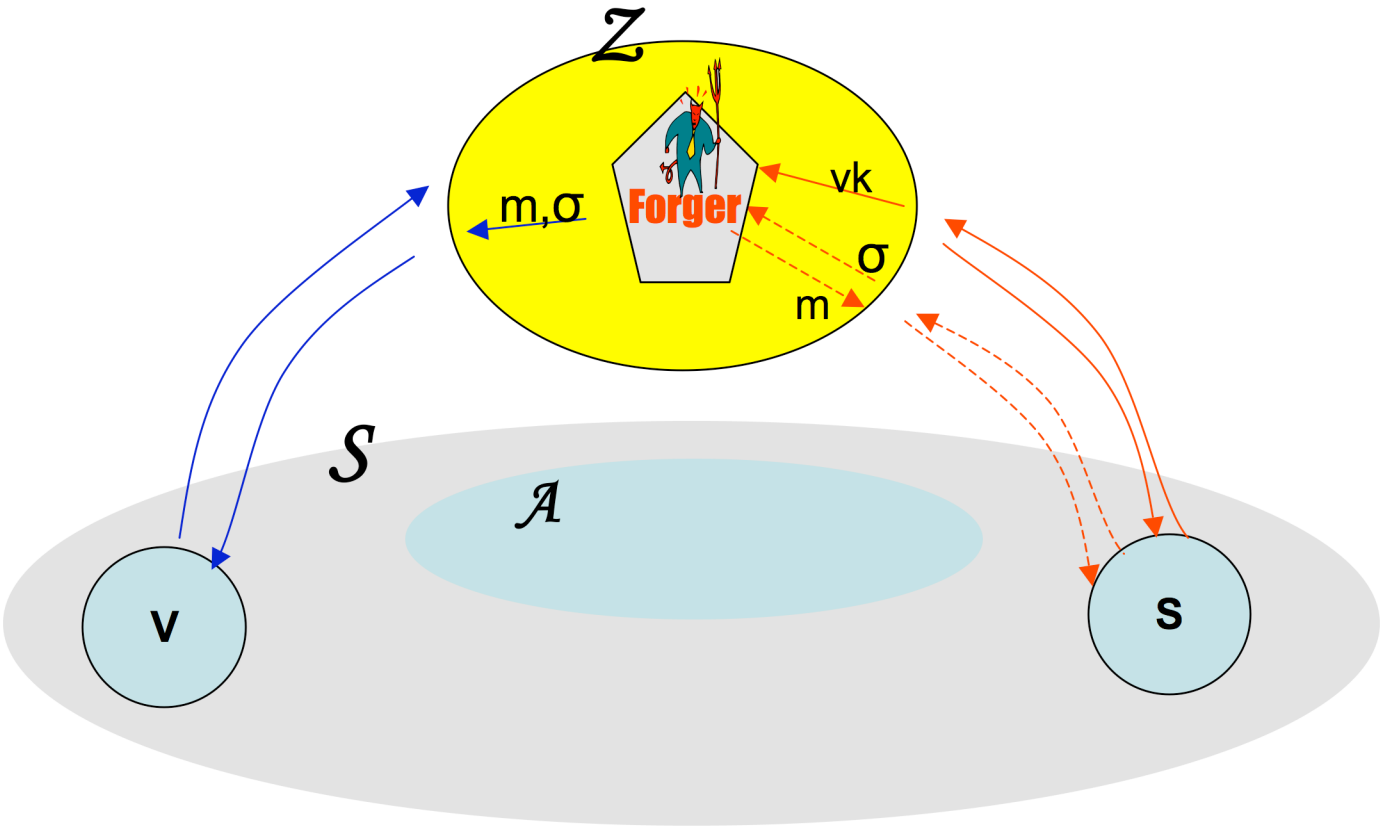
From Unforgeability to $\mathcal{F}^{\text{unforge}}$



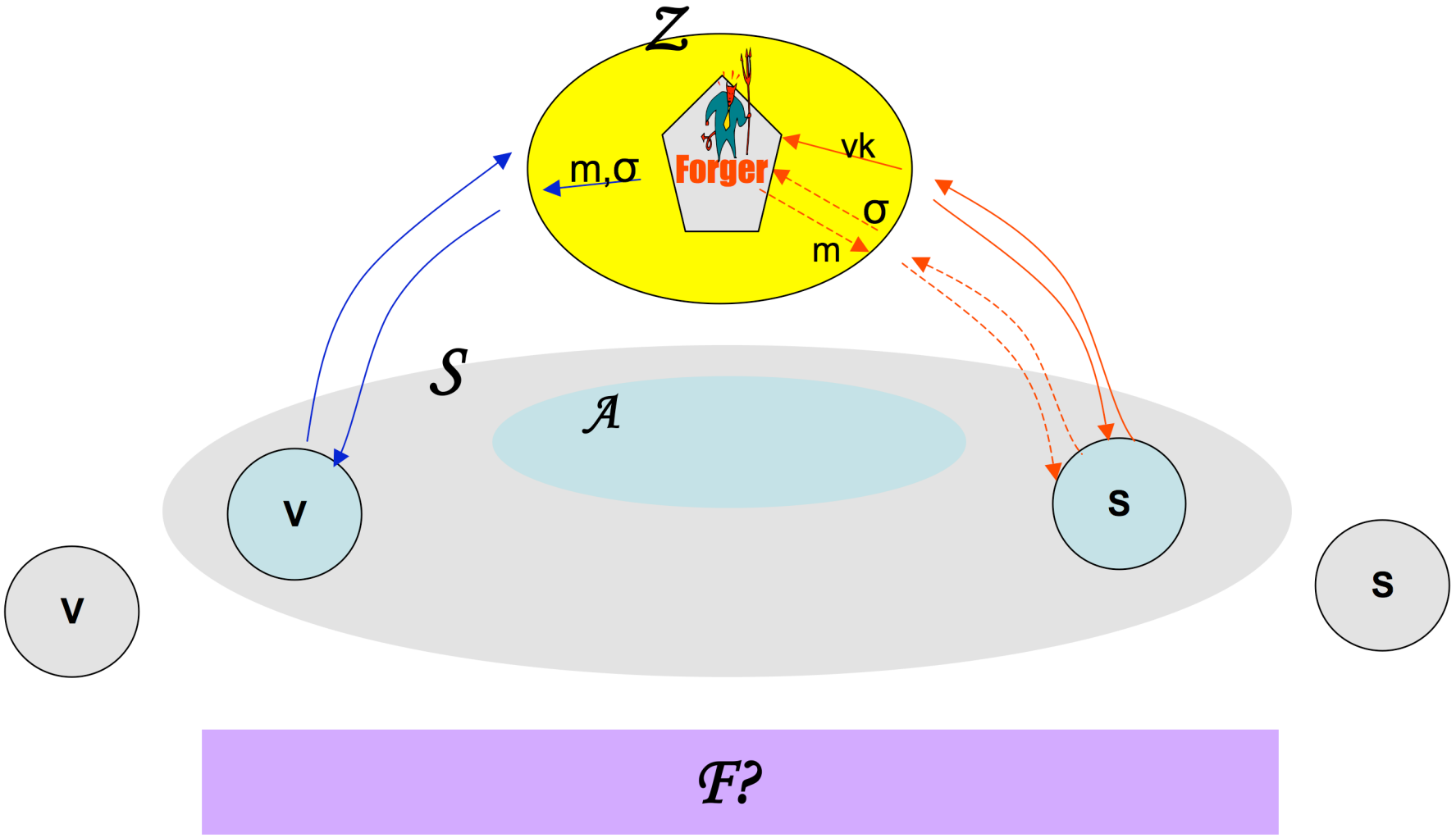
From Unforgeability to $\mathcal{F}^{\text{unforge}}$



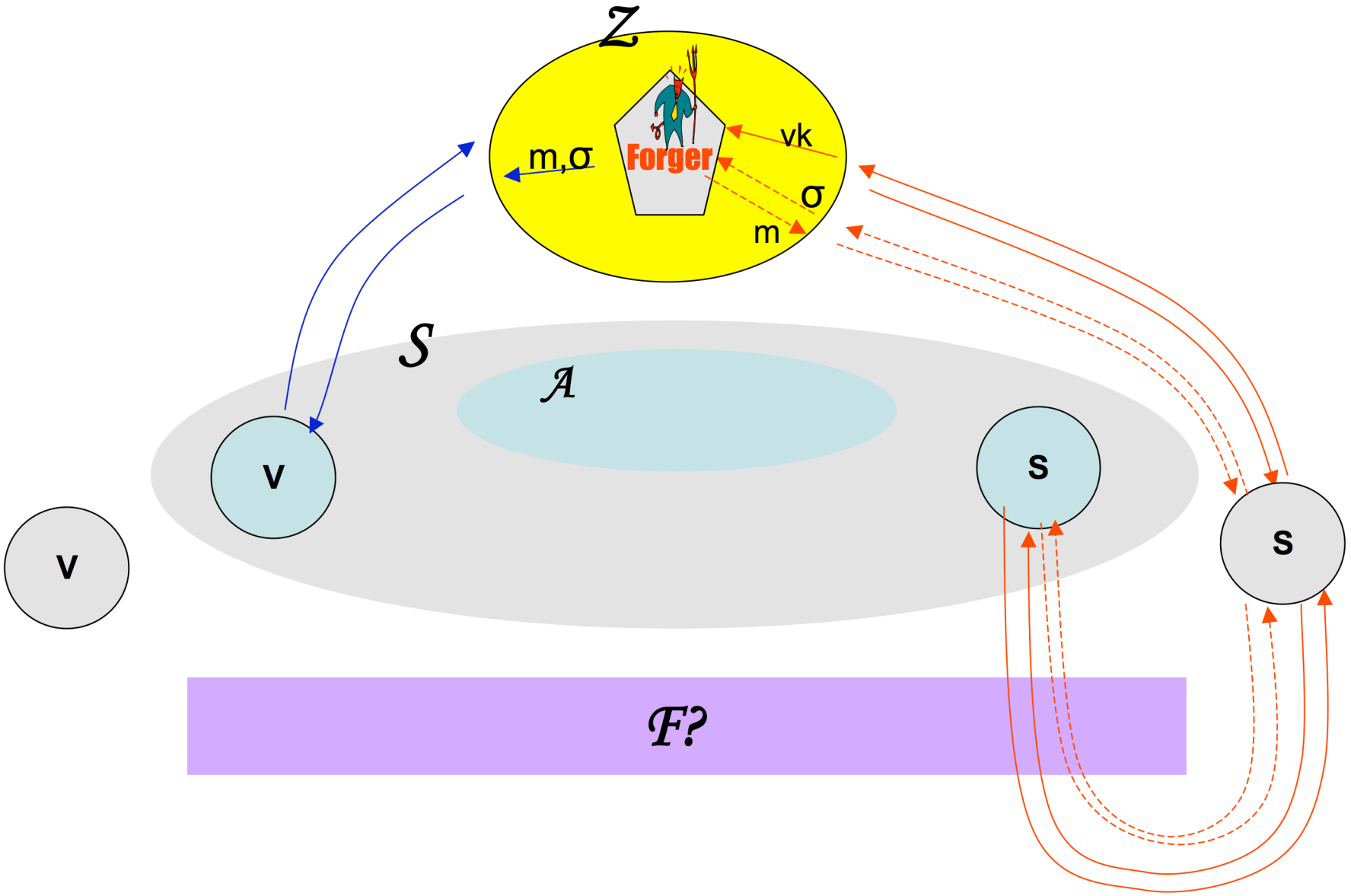
From Unforgeability to $\mathcal{F}^{\text{unforge}}$



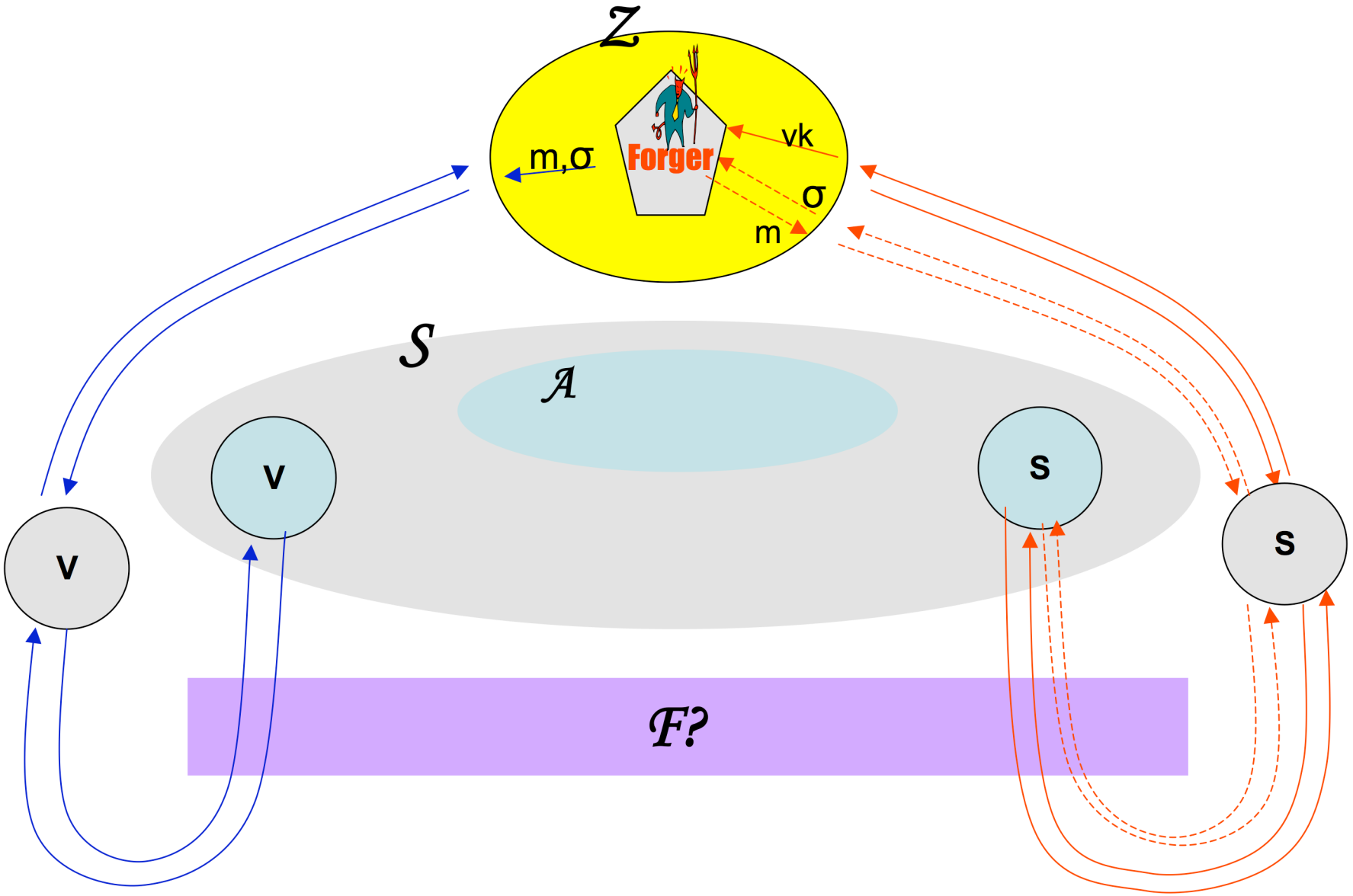
From Unforgeability to $\mathcal{F}^{\text{unforge}}$



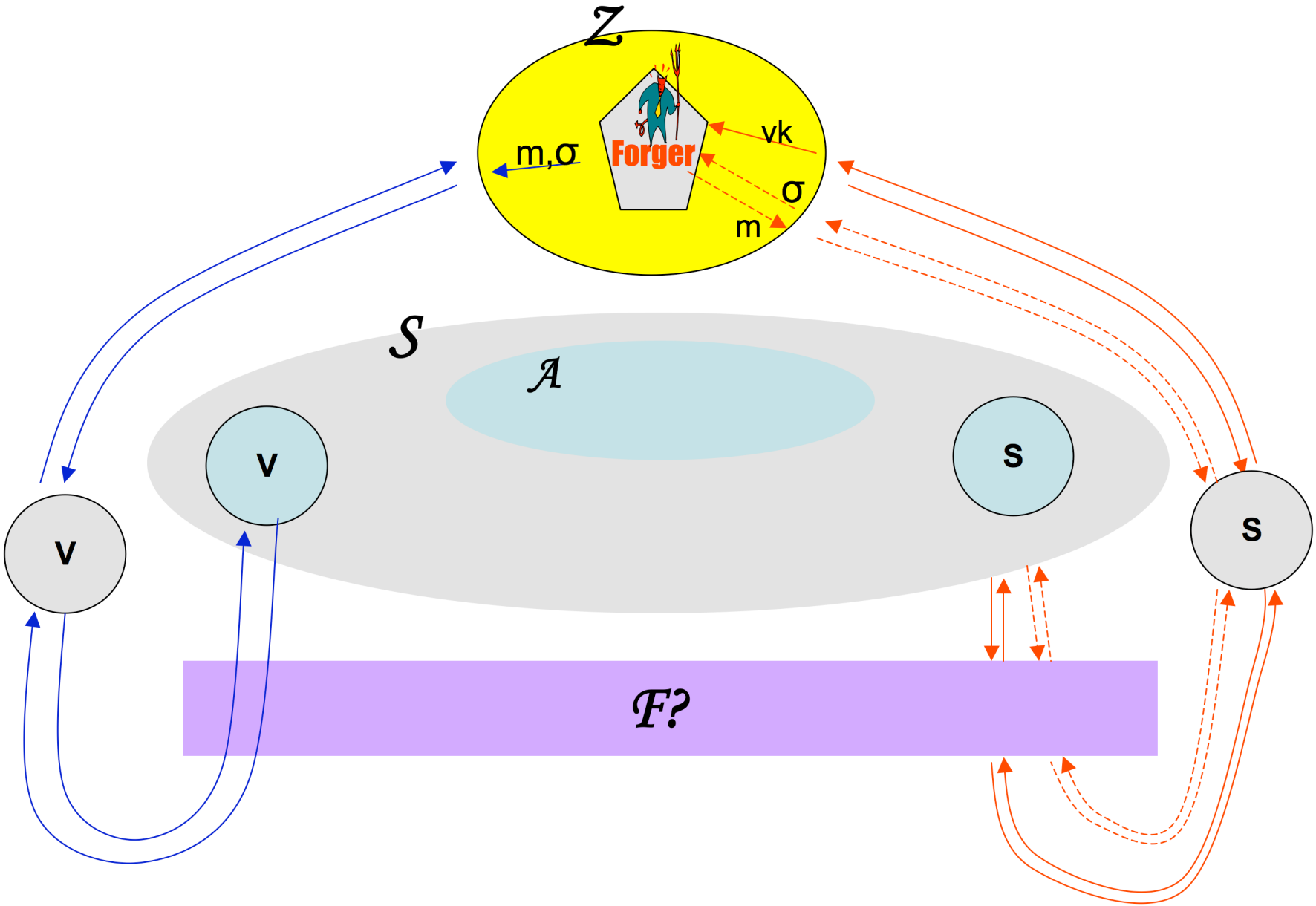
From Unforgeability to $\mathcal{F}^{\text{unforge}}$



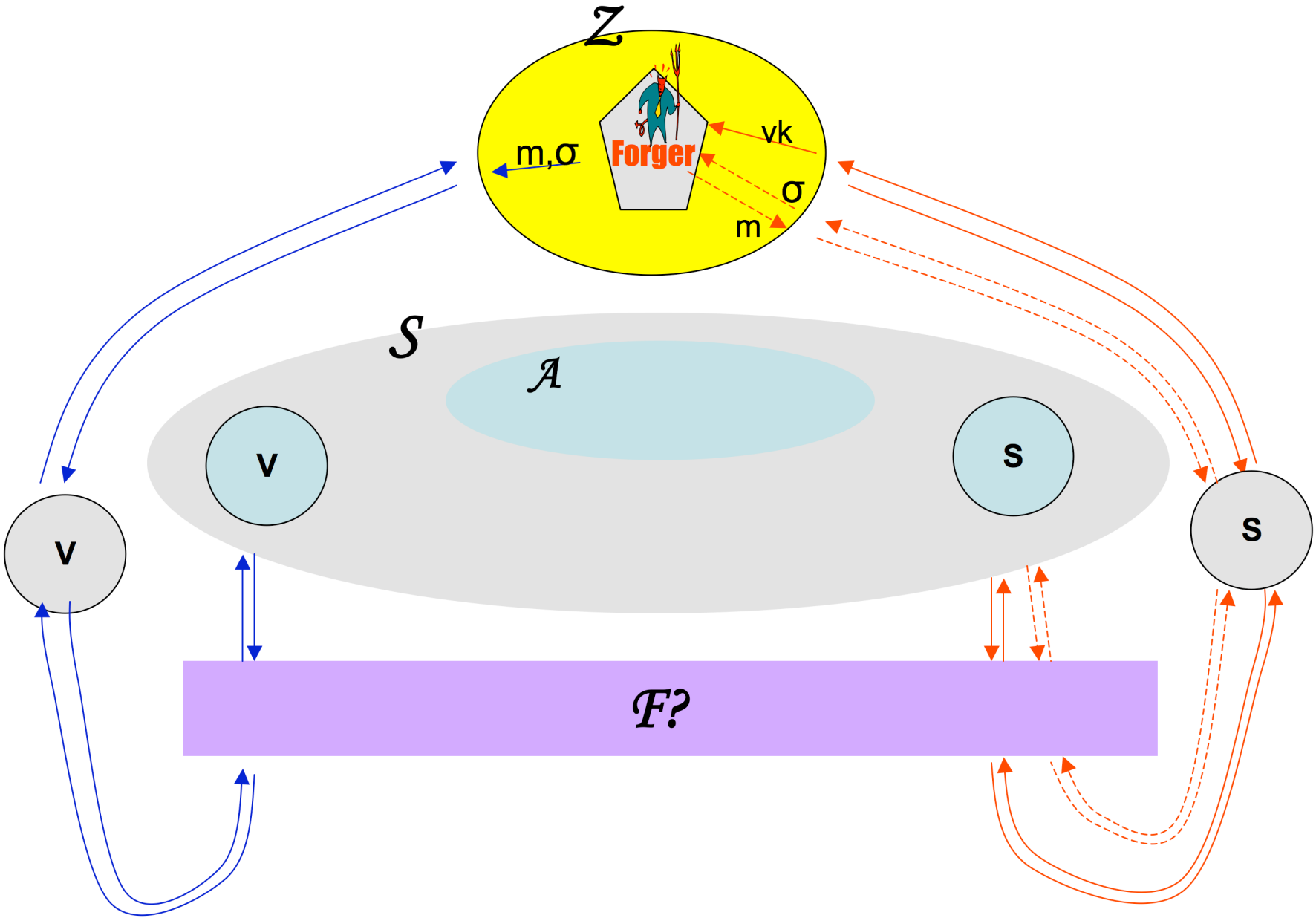
From Unforgeability to $\mathcal{F}_{\text{unforge}}$



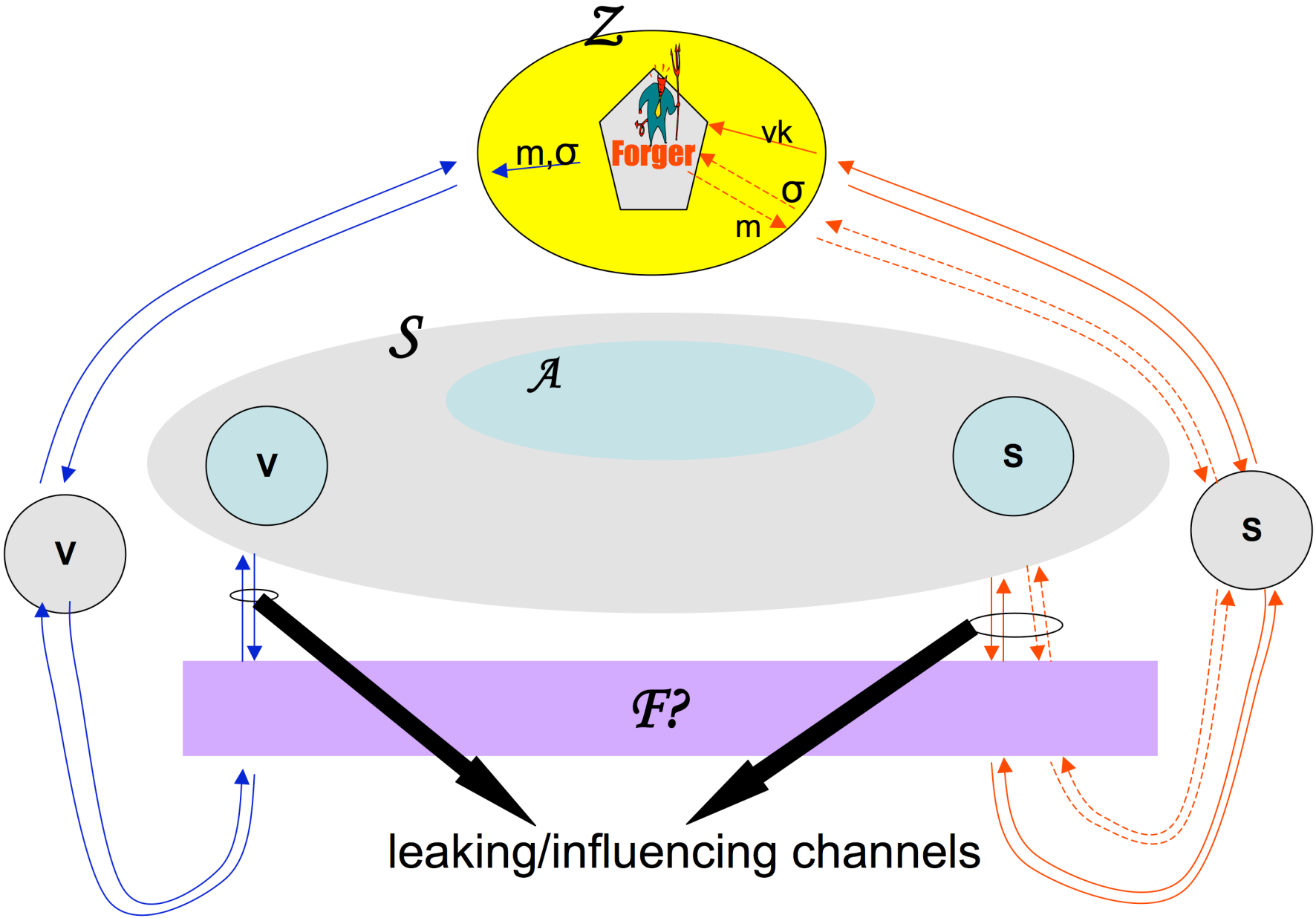
From Unforgeability to $\mathcal{F}_{\text{unforge}}$



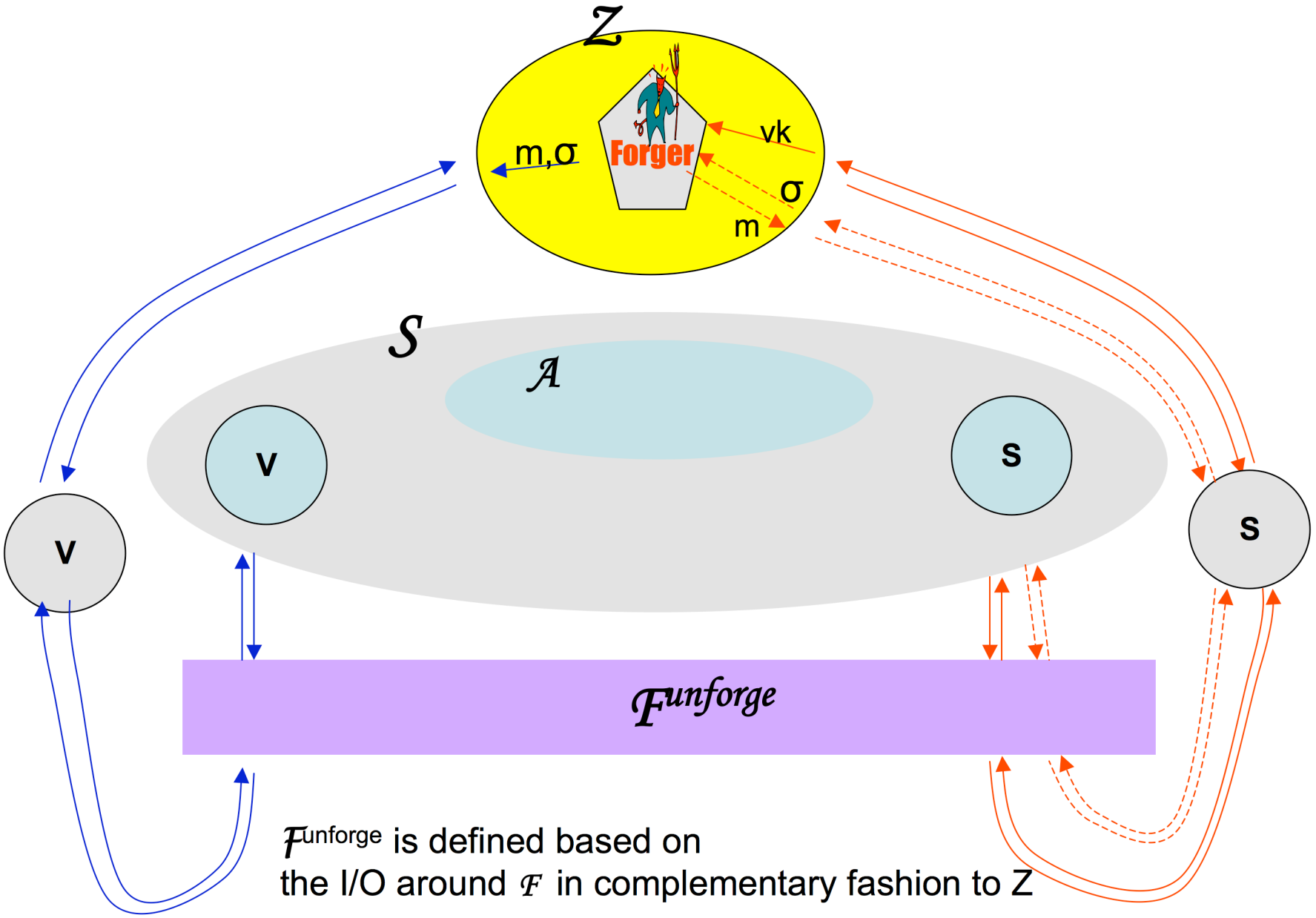
From Unforgeability to $\mathcal{F}_{\text{unforge}}$



From Unforgeability to $\mathcal{F}^{\text{unforge}}$



From Unforgeability to $\mathcal{F}^{\text{unforge}}$



$\mathcal{F}^{\text{unforge}}$ is defined based on
the I/O around $\mathcal{F}$ in complementary fashion to Z

General Approach

- Start with traditional def.
- Define environment around it.
- Form ideal $\mathcal{F}$ I/O language specifying influence/leaking channels.
- Obtain CLASS of functionalities that defeats the environment.

Sound and Fine-grain Specification of Ideal Functionalities

Juan Garay (*Bell Labs*)
Aggelos Kiayias (*University of Connecticut*)
Hong-Sheng Zhou (*University of Connecticut*)