Security in HTCondor 9.0

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

PATh PARTNERSHIP to ADVANCE THROUGHPUT COMPUTING

Slide from HTCondor Week 2020

“Forget what you know about HTCondor security. We changed it”

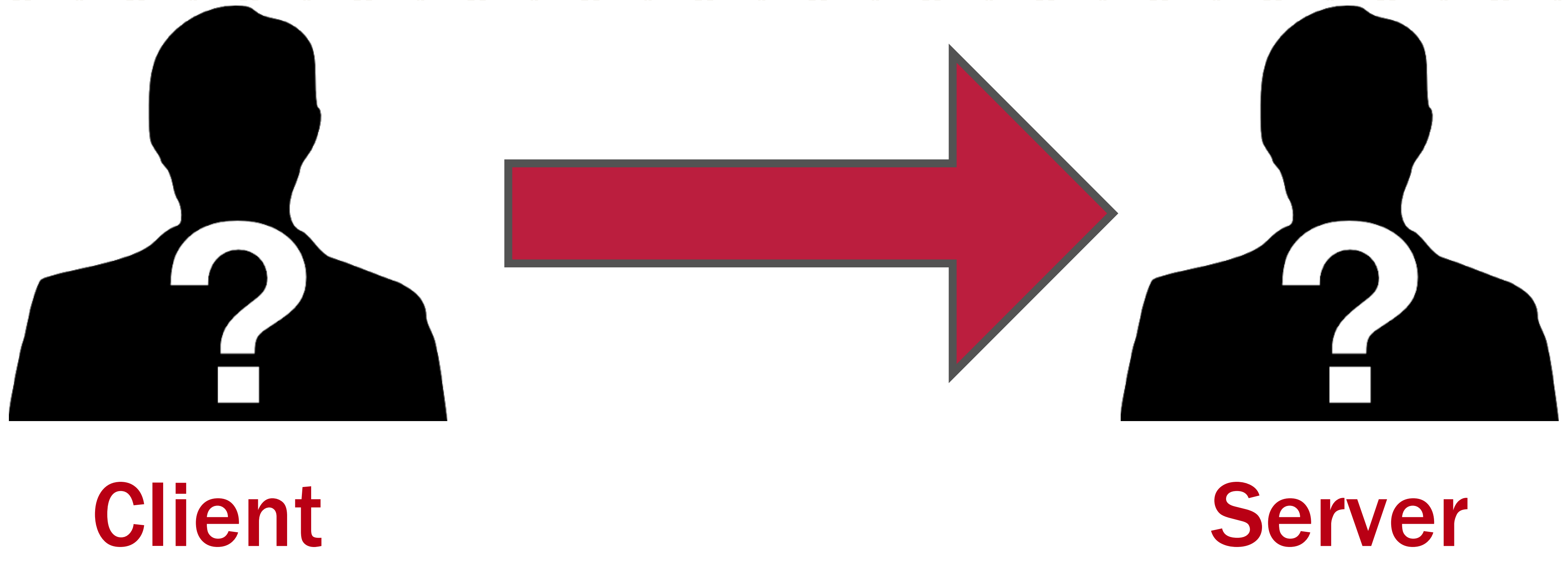
MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

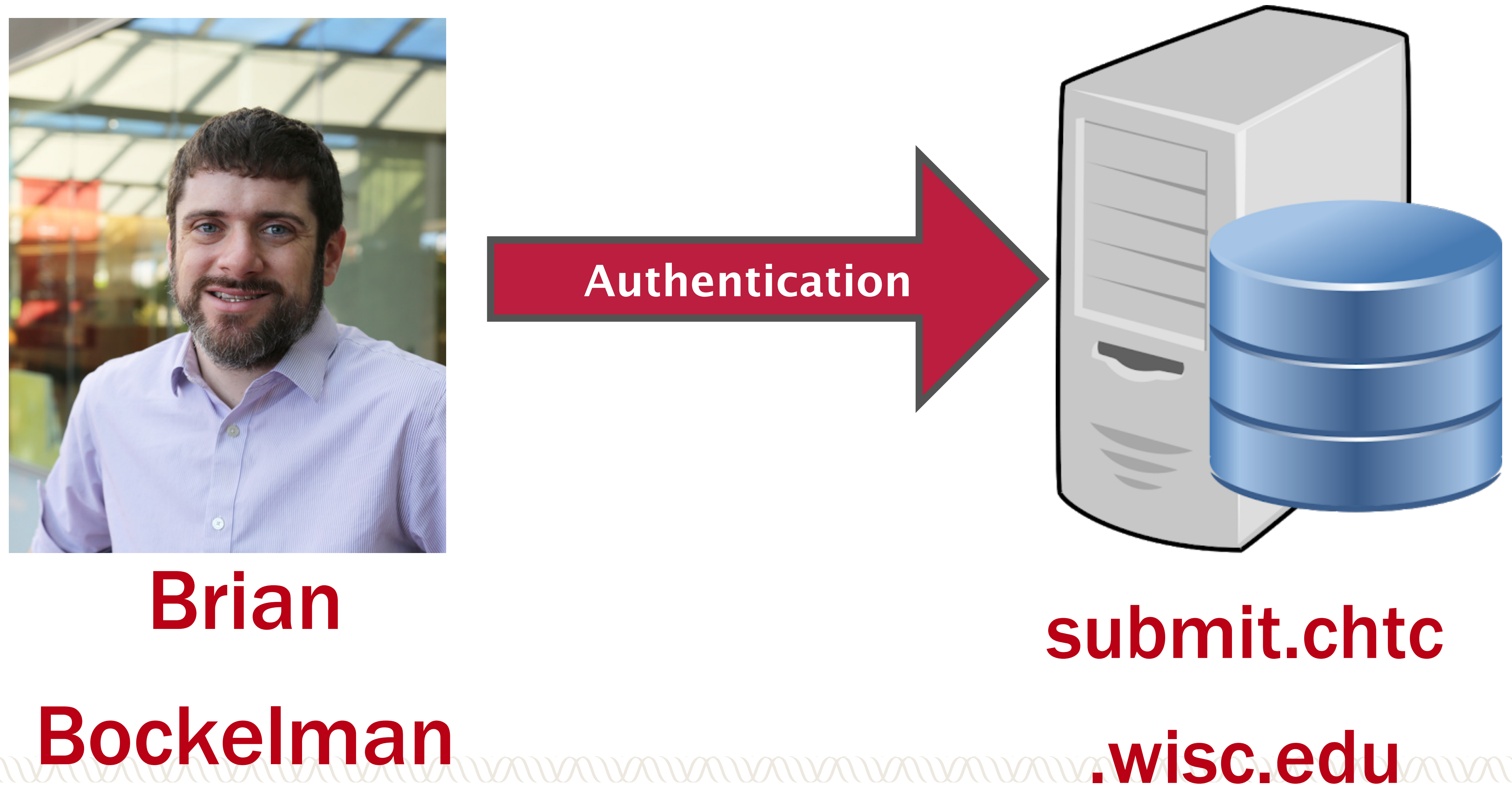
We Kept Working in 2021!

The HTCondor Authorization Model

What's the HTCondor Authorization Model?

1. Two entities make a connection over the network.
   - The two sides agree an a list of authentication methods to try.
2. The client and server authenticate with each other, establishing mutual identities.





The HTCondor Authorization Model

What's the HTCondor Authorization Model?

Two entities make a connection over the network.

- The two sides agree an a list of authentication methods to try.

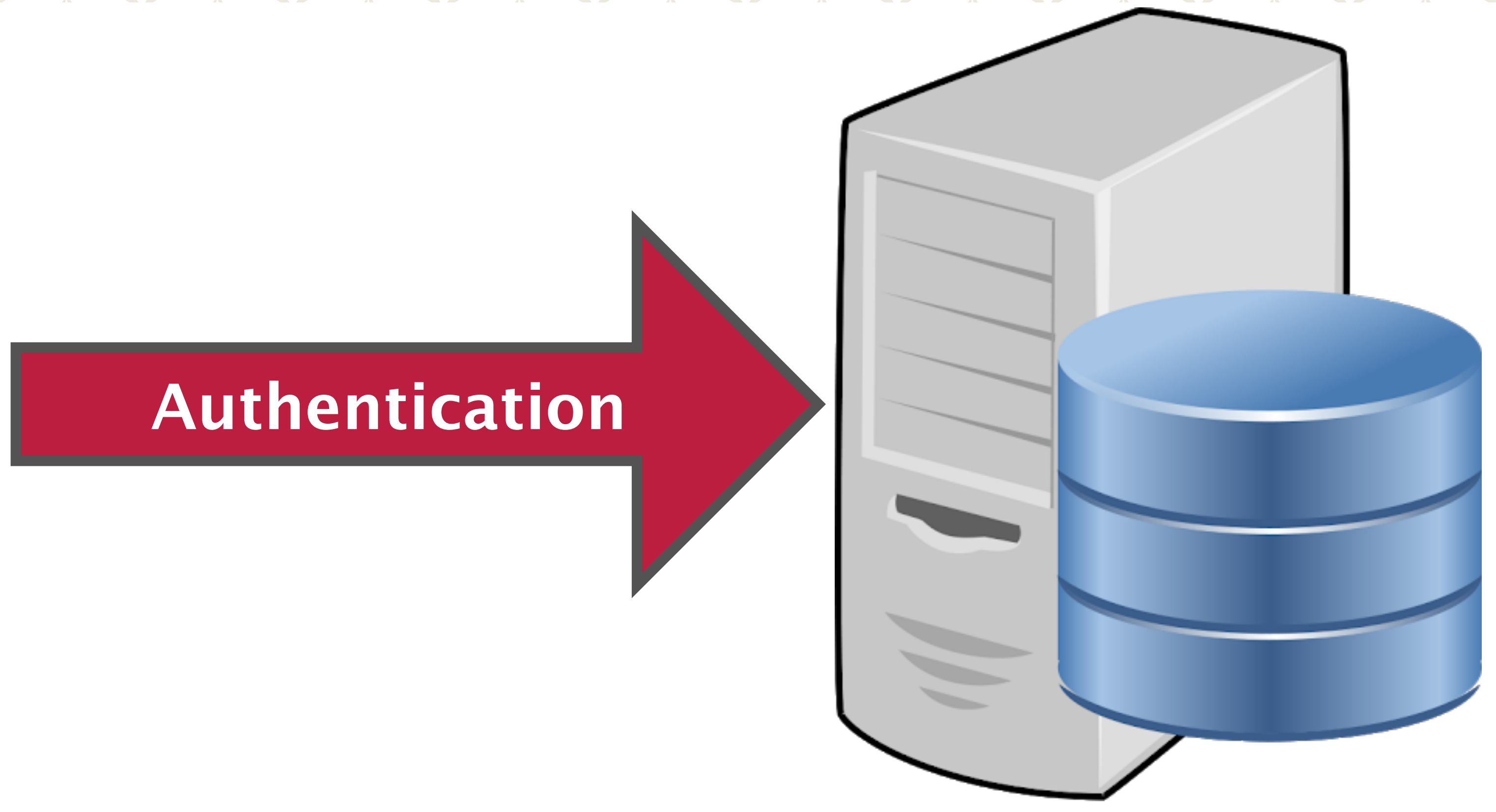
The client and server authenticate with each other, establishing mutual identities.

Each remote identity is mapped to a list of authorizations permitted.

Resulting authorizations are saved into a security session.

- Sessions have a unique identity, expiration, list of authorizations, and a private key.

Brian Bockelman



submit.chtc .wisc.edu

Session 1234
Remote User: Brian
Expiration: Next Monday
Authorizations: Read, Write
Private Key: ABCD

So, what’s new in HTCondor 9.0?

What’s new in HTCondor 9.0 in the HTCondor Authorization Model?

1. New authentication methods, **IDTOKENS** and SCITOKENS.
2. Changes in the **default authentication**.
3. Additional uses of auto-generated **sessions**.
4. Different model for **job identity**.

Additionally, the encryption and integrity were upgraded to modern algorithms. Not much to say in this talk besides “it won’t make your security people wince anymore”.

Topic 1:

The IDTOKEN

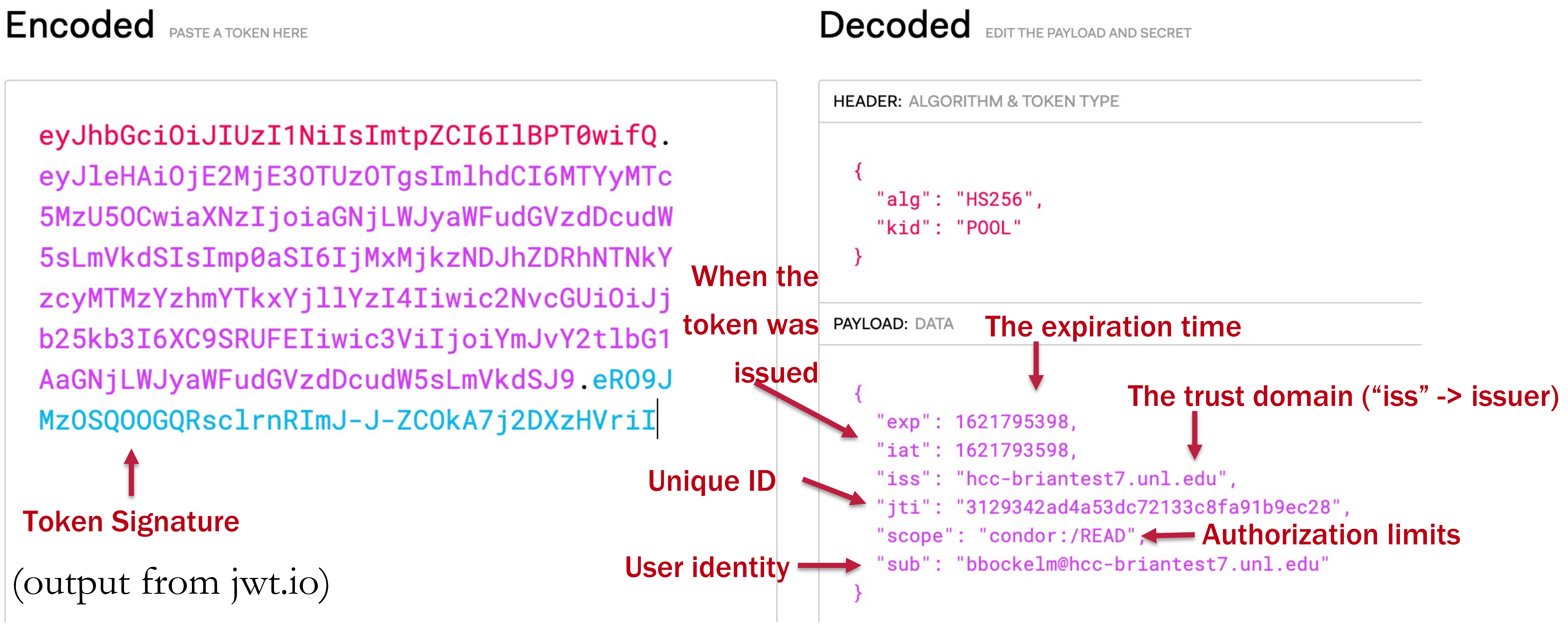
eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.eyJpYXQiOjE1ODk1NjYwOTEsImlzcyI6I
mNvbGxlY3Rvci5leGFtcGxlLmNvbSIsImp0aSI6ImQyODI1YjNhYTkyNzcyYWQ3ZmJi
NmNmMDNmZmI0ZmU2Iiwic3ViIjoiYnJpYW4uYm9ja2VsbWFuQGNvbGxlY3Rvci5le
GFtcGxlLmNvbSJ9.z8LUtjmqL_bqXTtUpC0-nXGflBfW3zI0JuB43S9MOGE

The IDTOKEN

An IDTOKEN is a token that can be used by a client prove possession of a shared secret – and hence establish an authenticated session.

- Each token is signed by a **signing key**. The server uses the signing key to verify the remote side has a valid token.
  - Consequence: IDTOKEN can only be used by a client. A daemon with only an IDTOKEN cannot be a server (but that's probably OK!)
- Each token contains an identity, an issued time, a unique ID as well as possible limits:
  - Expiration date.
  - Authorization limits. If present, these **reduce** the authorizations the server configured – does NOT add to them.

Big secret: IDTOKENS are JWTs



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.
eyJleHAiOjE2MjE3OTUzOTgsImlhdCI6MTYyMTc
5MzU5OCwiaXNzIjoiaGNjLWJyaWFudGVzdDcudW
5sLmVkdSIsImp0aSI6IjMxMjkzNDJhZDRhNTNkY
zcyMTMzYzhmYTkxYjllYzI4Iiwic2NvcGUiOiJj
b25kb3I6XC9SRUFEIiwic3ViIjoiYmJvY2tlbG1
AaGNjLWJyaWFudGVzdDcudW5sLmVkdSJ9.eR09J
MzOSQOOGQRsclrnRImJ-J-ZCOkA7j2DXzHVriI
```

Token Signature

(output from jwt.io)

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "kid": "POOL"
}
```

PAYLOAD: DATA

```
{
  "exp": 1621795398,
  "iat": 1621793598,
  "iss": "hcc-briantest7.unl.edu",
  "jti": "3129342ad4a53dc72133c8fa91b9ec28",
  "scope": "condor:/READ",
  "sub": "bbockelm@hcc-briantest7.unl.edu"
}
```

When the token was issued

The expiration time

The trust domain ("iss" -> issuer)

Unique ID

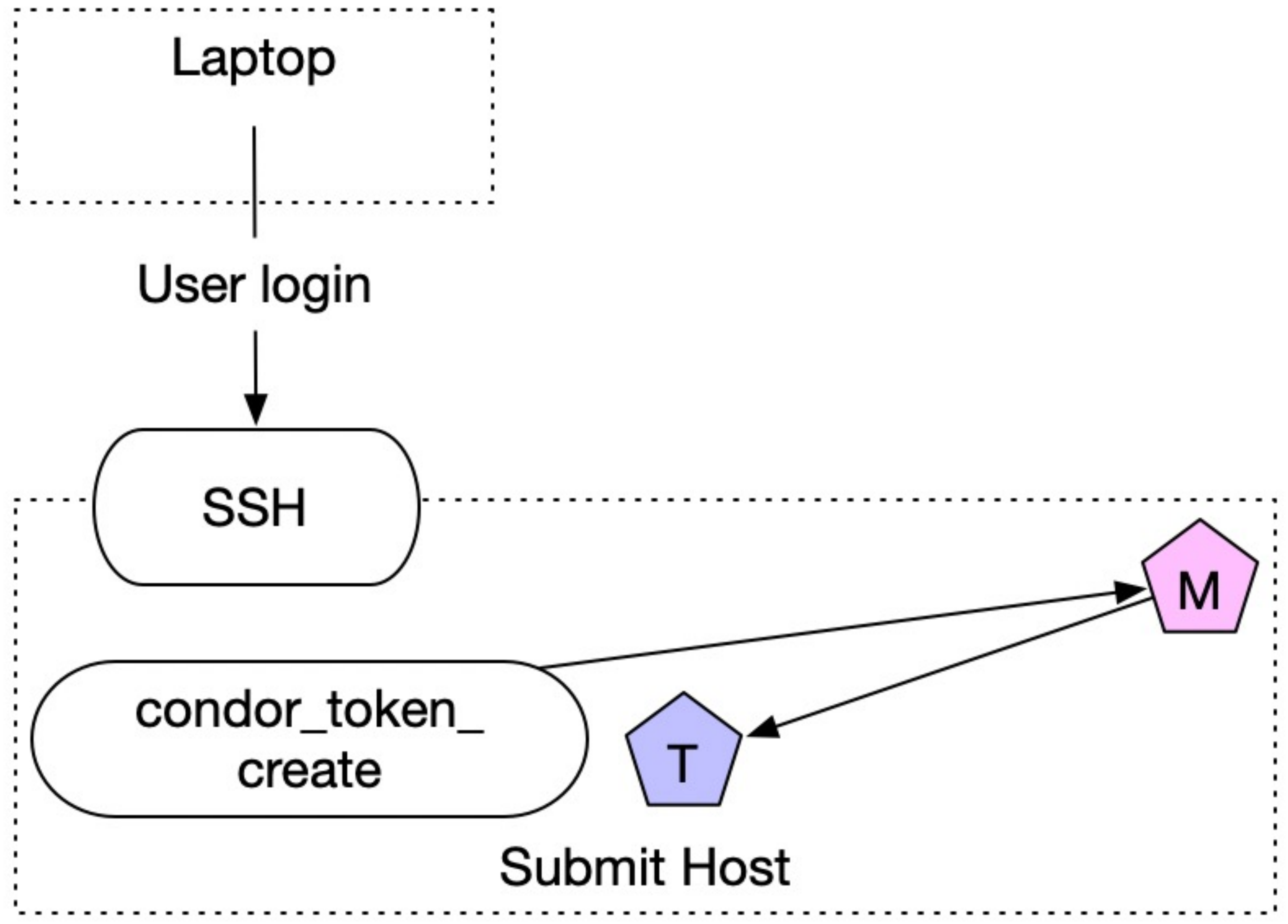
Authorization limits

User identity

Bootstrapping Trust – Creating an IDTOKEN

Anyone who can read the signing key can issue any token they want using **condor_token_create**.

```
$ sudo condor_token_create \
      -identity brian.bockelman@collector.example.com \
      -lifetime 3600 \
      -authz READ -authz WRITE
```

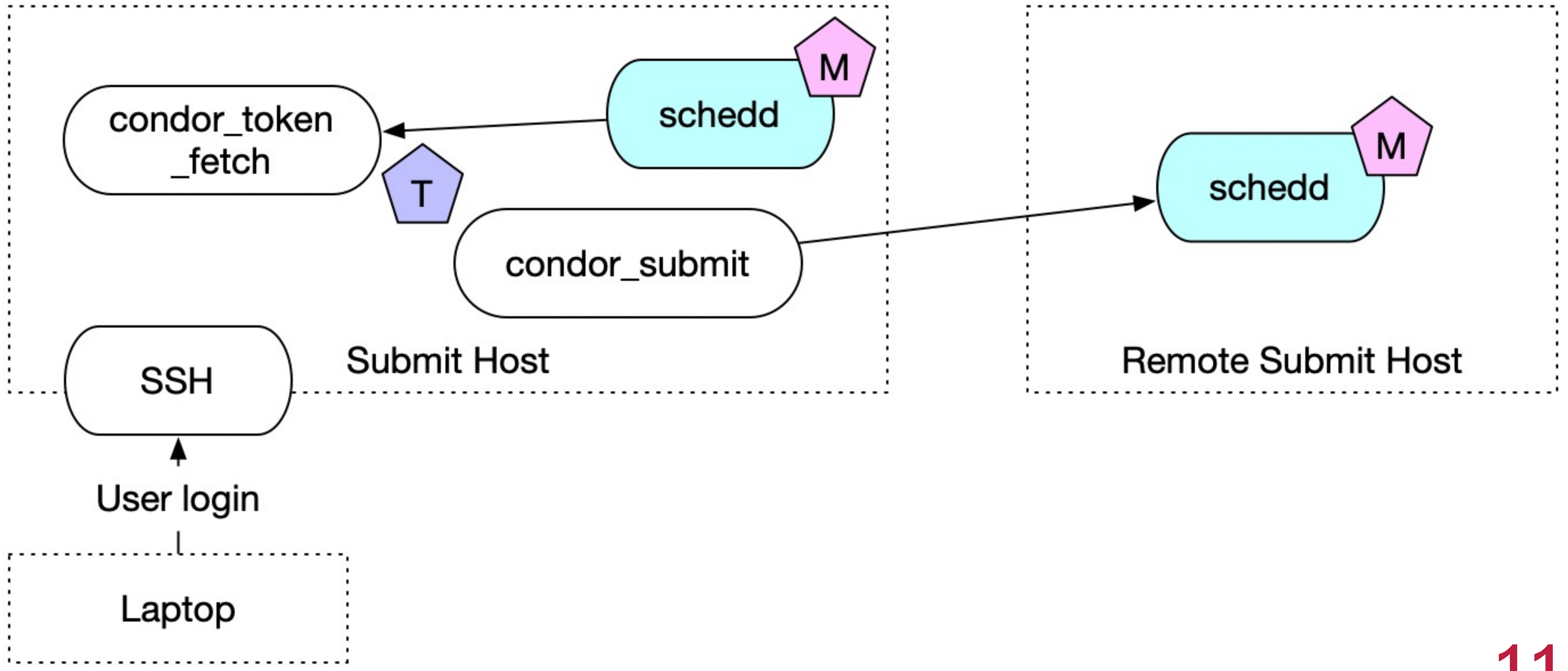


```
eyJhbGciOiJIUzI1NiIsImtpZCI6IlBPT0wifQ.eyJleHAiOjE1ODk4Mjk4MzUsImlhdCI6MTU4OTgyNjI
zNSwiaXNzIjoiY29sbGVjdG9yLmV4YW1wbGUuY29tIiwic2NvcGUiOiJjb25kb3I6L1JFQUQgY29u
ZG9yOi9XUklURSIsInN1YiI6ImJyaWFuLmJvY2tlbG1hbkBjb2xsZWN0b3IuZXhhbXBsZS5jb20if
Q.NxOw5f9GsmGgwV0TezisZwmtqRbRuGHvj8G1r5esdLI
```

Fetching an IDTOKEN

Does authentication work now – but you need to squirrel away an IDTOKEN for future use? **condor_token_fetch** to the rescue!

- This tool authenticates with a daemon and asks the daemon to sign a token on behalf of the user's identity. Resulting identity is identical to authenticated ID.
- **Use case:** I have an SSH login to a local schedd but want to remotely submit to a schedd in the same trust domain.

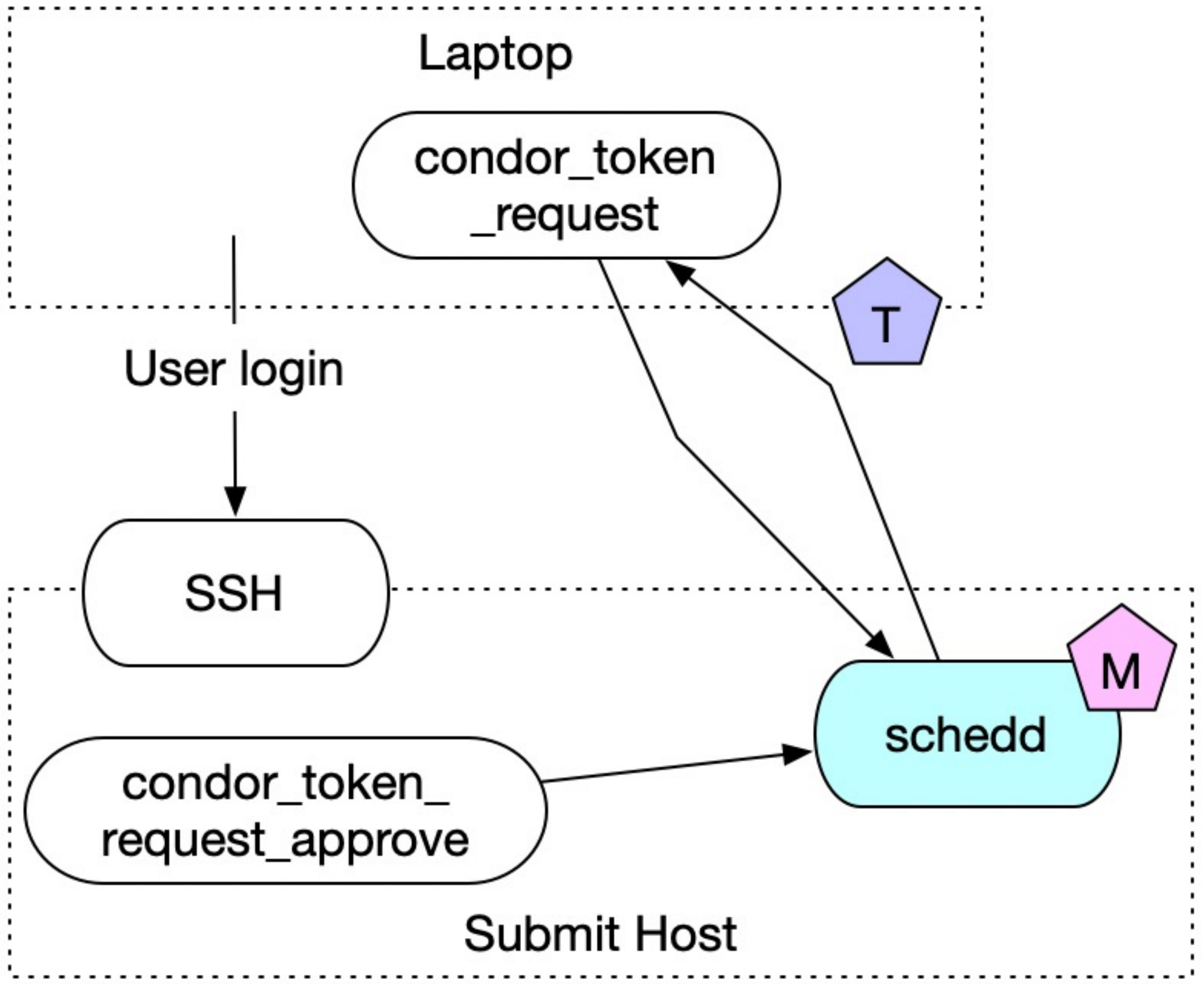


Requesting an IDTOKEN

Want to get an IDTOKEN on a machine without authenticating?

- **condor_token_request** allows an anonymous user to request a token for an arbitrary identity X.
  - The token request can be approved either by an admin or a user authenticated as X.
  - Anyone can ask. Few can approve!
- **Use case**: I have an SSH login on a schedd and want to start submitting jobs from my laptop.
  - **Solution**: Request a token from my laptop; login to the submit host and approve the request.
- ***DO NOT COPY/PASTE TOKENS.*** Instead, use **condor_token_request**!

The startd, master, and schedd will automatically request tokens from the collector if authentication fails.



Gotcha: to work, the client needs to trust the server – typically, this implies SSL authentication (which is tricky to setup). Look forward to new tricks in 9.1.x...

Topic 2: New Defaults

Prior to 9.0, the default authentication mechanism was **host-based**: trust anyone that appears to be coming from a certain IP address.

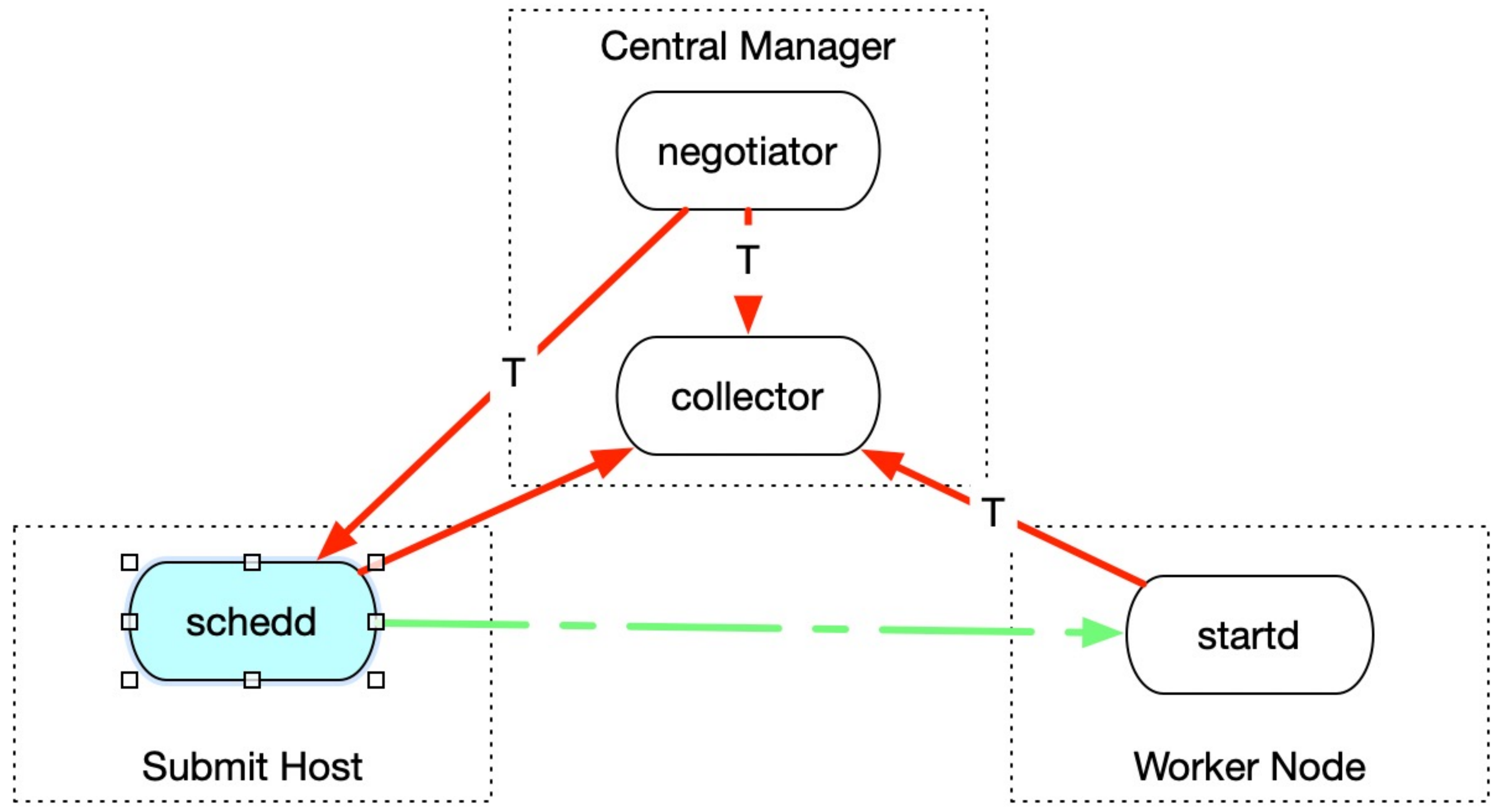
- Not what you wanted: the collector couldn't tell the difference between the 'real' schedd and any user logged in via SSH.

In 9.0, there **no host-based authentication is enabled by default**.

- Default authentication options are NTSSPI (Windows-only), FS, IDTOKENS, KERBEROS, GSI, SCITOKENS, SSL.
  - Note: CLAIMTOBE is automatically added for READ sessions.
- Recommendation: share a pool password among your trusted hosts and generate IDTOKENS for user sessions.
- This is exactly what the **`get_htcondor`** script does for you…

Further, encryption and integrity-checking will occur by default.

Topic 3: Sessions in HTCondor



The startd trusts any negotiator trusted by its collector!

+

The negotiator trusts any schedd in the collector.

=

The startd trusts any schedd in its collector.

For a few years, HTCondor has had "match password" security. In this case,

- The startd generates a capability, T, and sends it to the collector in its ClassAd.
  - **Anyone with T is allowed to start jobs on the startd.**
- The negotiator gets T from the collector because the collector trusts the negotiator.
- The schedd gets T as part of the 'match' created by the negotiator.
  - Hence the name "match password".

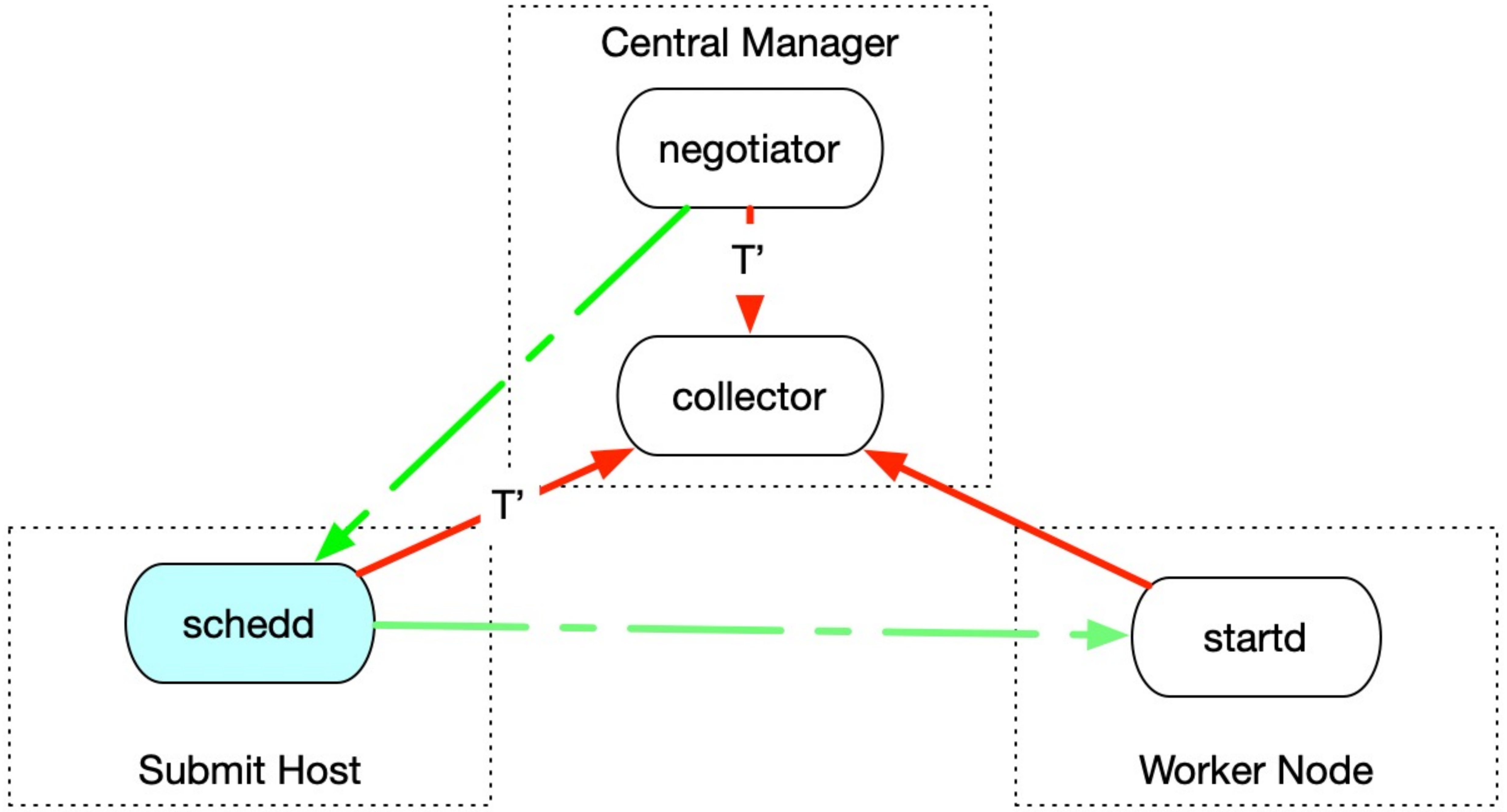
Extending Trust in the Collector

Starting in 8.9.x, the schedd also generates a session, T', and sends it in its ClassAd.

- The schedd trusts the collector only gives T' to trustworthy negotiators.
- Any client with T' is allowed to be a negotiator for the schedd.

This is also used for user-level flocking: the schedd creates a special session only good for negotiating for a single user, then advertises it to a remote collector using the user's token.

- Result: user can flock to a given collector even without trust between the schedd admin and the collector.



Topic 4: Identities in HTCondor

A session is used to establish an HTCondor identity. This is not the only place identities are used! The most complex example is a job execution. **Let's count the identities**:

1. HTCondor identity of the client who submitted the job.
2. **The Unix identity on the submit host where the input files are saved.**
3. The accounting identity whose fairshare is used when jobs are run.
4. The Unix identity on the execution host where the job is run.

Prior to HTCondor 9.0, (2) was the primary identity. For example, a client can remove a job if and only if your calculated value of (2) is equal to the job's value of (2).

- The difference between (1) and (2) was academic: (2) was generated by truncating (1) at the first '@' character.
  - Quick Question: what happens if bbockelm@gmail.com and bbockelm@cern.ch both wanted to submit to our pool?

A new Owner is in town…

In 9.0, we can make the HTCondor authenticated identity the primary identity (off by default):

- Two users with the same username but different domain will not be able to manage each other's files.
- The Unix account becomes an implementation detail.
- A Unix account need not exist on the schedd for the user:
  - If the user's domain does not match the schedd's UID_DOMAIN setting, the Unix owner becomes the special username **nobody**.
  - Hence, the job's identity can be decoupled from the Unix username; may be especially helpful in the case of web portals where the web users may not be allocated a Unix account.

Some work is left in 9.1 before enabling this by default; currently hidden behind the **`USER_IS_THE_NEW_OWNER`** configuration variable.

Looking to the Future

IDTOKENS provides a simple and effective mechanism for securing a HTCondor pool.

- Unlike other mechanisms, we provide tools to help bootstrap IDTOKENS auth!

Because of this, we disable host-based auth by default and enable encryption/integrity.

- More work to setup a HTCondor pool – we need at a minimum to give each host a shared secret. But we also provide tools for this too – see Todd Miller's Wednesday AM talk!

What's left to do in 9.1? The biggest hassle (and security risk!) is the distribution of the signing keys:

- It's not possible to send commands (like fetch_log or defrag) to StartD's with IDTOKENS; this is why we currently distribute the signing keys everywhere. We would like each StartD to setup a management session and send it to the collector.
- Bootstrapping with token requests requires SSL auth to be setup, which is incredibly tricky. Looking into ways we can do auto-generation of SSL certs and SSH-like trust-on-first-use for clients.

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

morgridge.org

This material is based upon work supported by the National Science Foundation under Grant No. 1738962 and 2030508. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

FEARLESS SCIENCE